Final Technical Report PFTR-1061-79-1
Contract MDA903-78-C-0127
ARPA Order No. 3488
January, 1979

ADAPTIVE INFORMATION SELECTION FOR SUPPORT OF C-3 FUNCTIONS IN SMALL TACTICAL UNITS: INITIAL SYSTEM DEVELOPMENT AND SOFTWARE DOCUMENTATION

AZAD MADNI
RANDALL STEEB
DAVID RUBIO
ROBERT FARNUM

Prepared For:

ADVANCED RESEARCH PROJECTS AGENCY
Cybernetics Technology Office
1400 Wilson Boulevard
Arlington, Virginia 22209

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470

NOTES

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any office of the United States Government.

Final Technical Report PFTR-1061-79-1
Contract MDA903-78-C-0127
ARPA Order No. 3488
January, 1979

ADAPTIVE INFORMATION SELECTION FOR SUPPORT OF C-3 FUNCTIONS IN SMALL TACTICAL UNITS: INITIAL SYSTEM DEVELOPMENT AND SOFTWARE DOCUMENTATION

AZAD MADNI
RANDALL STEEB
DAVID RUBIO
ROBERT FARNUM

Prepared For:

ADVANCED RESEARCH PROJECTS AGENCY
Cybernetics Technology Office
1400 Wilson Boulevard
Arlington, Virginia 22209

PERCEPTRONICS
6271 VARIEL AVENUE • WOODLAND HILLS • CALIFORNIA 91367 • PHONE (213) 884-7470

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: PFTR-1061-79-01	2. GOVT ACCESSION NO.: AD-A145 761
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Adaptive Information Selection for Support of C-3 Functions in Small Tactical Units: Initial System Development and Software Documentation.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Azad Madni, Bob Farnum, Randall Steeb, David Rubio	8. CONTRACT OR GRANT NUMBER(s): MDA-903-78-C-0127
9. PERFORMING ORGANIZATION NAME AND ADDRESS: PERCEPTRONICS, INC. 6271 Variel Avenue Woodland Hills, CA 91367	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: ARPA Order No. 3488
11. CONTROLLING OFFICE NAME AND ADDRESS: DARPA-Cybernetics Technology Office 1400 Wilson Blvd. Arlington, VA 22209	12. REPORT DATE: January 1979
	13. NUMBER OF PAGES:
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Adaptive Models	Information Management Aid
Computer-Aided Decisions	Information Selection
Decision Aid	Information Value
Multi-Attribute Utility	Man-Machine Interaction

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the development of the Adaptive Information Selection (AIS) System for Support of Information Management Functions in small tactical units. The report includes a description of the simulation system design and software functions. Included also is a user's guide for running the AIS system simulation and details of the transfer of the AIS system to the Tactical Combat Operations System (TCO) simulation on the Marine Tactical Command and Control Systems (MTACCS) Interim Test Facilty. The AIS system is based on a multi-attribute evaluation (MAE) of an individual user's preference structure or

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

policy. The program is designed to capture the information selection strategy using a training algorithm based on pattern recognition techniques. In the automatic mode, the AIS system supplies the TCO user with messages on the basis of his individual selection strategy. In this way, it provides an aiding channel to the TCO message routing and prioritization function. The results of the AIS system preliminary evaluation by TCO personnel were encouraging.

1. SUMMARY

1.1 Overview

This report, covering the period of January 1978 through December 1978, describes the initial development of the Adaptive Information (AIS) system for the Marine Corps Tactical System Support Activity (MCTSSA). The Aid system has been designed as an information management aid for the Tactical Combat Operation (TCO) system simulation on the Marine Tactical Command and Control Systems (MTACCS) Interim Test Facility. The first part of this report includes a description of the AIS system model, the AIS system design, the principle software functions, and a user's guide for running the AIS system simulation. Also included in the latter part of the report are preliminary AIS evaluation results and details of the planned transfer of the AIS system software to the TCO simulation, scheduled for January 1979 through December 1979.

1.1.1 Background. The AIS system is based on an adaptive multi-attribute utility model for information selection. This concept was developed and demonstrated by Perceptronics under ARPA Contract No. MDA903-76-C-0241 (Freedy, Davis, Steeb, Samet and Gardiner, 1976). In this effort, the adaptive decision model was applied to and evaluated on a simulated ASW submarine tracking task. Improvements in speed and efficiency of task performance were noted with use of the aid (Samet, Weltman and Davis, 1976). Successful demonstration of the prototype information selection system led to proposals for its use in a variety of other military C3 situations.

In the current program the adaptive multi-attribute model was modified to provide a real-time information aid designed to increase the combat effectiveness of small military maneuver units. The target unit was the Marine battalion, and the work was performed with the close cooperation of MCTSSA,

Camp Pendleton, California. Primary points of contact were Col. A.I. Warczakowski and Lt. Col. Wickens. MCTSSA provided battalion test scenarios, access to their Interim Test Facility, Marine personnel and test facility support, and AIS system requirements analysis.

1.1.2 Objectives. Specific objectives of the one year program reported here included the following:

(1) Analysis of TCO operations and identification of the specific functions to be performed by the AIS.
(2) Development of the overall structure of the AIS model in the context of the TCO application: specification of principle functions, formulation of data structures, and definition of model attributes accessible from the simulated TCO messages resident in the data base of the MTACCS Interim Test Facility.
(3) Development of AIS working software, and "gage test" demonstration of AIS system aiding for message distribution and prioritization in a simulated C3 scenario representative of the TCO scenario on the MTACCS Interim Test Facility.
(4) Review and analysis of the evaluation measures planned by the Marine Corps Tactical Systems Support Activity (MCTSSA) for the coming MTACCS Interim Test Facility exercises in the light of AIS evaluation procedures.
(5) Establishment of design guidelines for transfer of the AIS system software into the TCO simulation on the MTACCS Interim Test Facility.

1.2 Technical Approach

1.2.1 Rationale. Technical advances have led to increases in the speed, mobility, and destructive power of military operations. The amount and

transfer rate of information has increased accordingly. Information must be processed more efficiently and more effectively for commanders to make tactical decisions responsive to the rapidly changing succession of events. To meet this need, new computer-based systems for command, control, and communications (C3) are being developed and implemented. These systems are intended primarily to aid in the collection, processing, and utilization of different types and amounts of military data. The overall process is cyclic -- as information is being used, other information is being processed, and new information is being sought and collected. The dynamics of information flow are, therefore, of critical importance and must be constantly monitored and directed.

The consensus concerning current computer-based military systems for C3 operations is that they have increased the rate and density of information flow to such an extent as to overwhelm a commander and his staff. New C3 techniques are required to control information flow so as to best match system capability with human characteristics in the man-computer interaction. Review of previous research suggests that a significant step in this direction would be to individualize and automate information selection. This would allow each system user continuously to obtain information that is both relevant and timely with regard to his individual processing characteristics and immediate decision making needs. Considering the large number of interrelated users in a typical C3 system such as TCO, the effect on total system performance would be to increase throughput substantially while also improving decision making quality.

1.2.2 System Concept. The basic concept of the model-based selection system is illustrated in Figure 1-1. The message universe includes all information potentially available to the recipient, or system user. In the manual mode, the recipient continually selects messages in accord with some selection stragegy. A strategy represents individual preferences for infor-

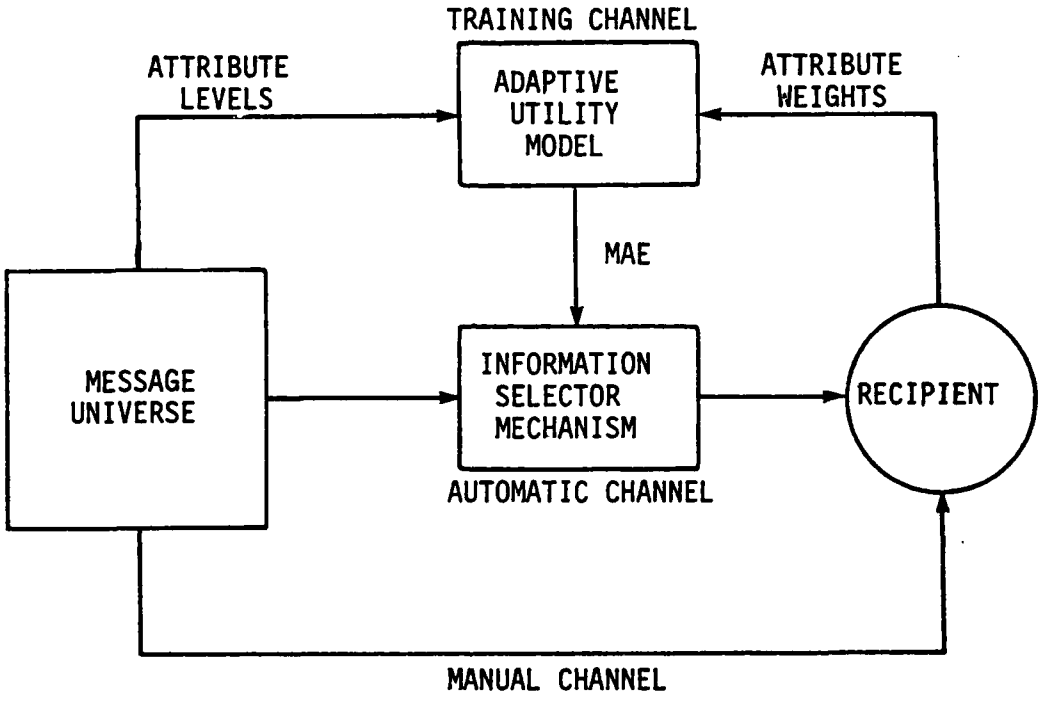


FIGURE 1-1. INFORMATION SELECTION SYSTEM CONCEPT

mation in response to situational needs. In the automatic mode, an adaptive information selection mechanism automatically supplies the user with information on the basis of his or her individual selection strategy.

The factors which characterize an individual's strategy are incorporated in an adaptive multi-attribute utility model. In this model, incoming information is decomposed into measurable attributes. Attribute levels of a message are defined by vectors which include both situational requirements and source characteristics. The subjective weight, or utility, that the user places on each attribute is estimated on-line, by an adaptive technique, as the user manually selects information. This is done during a system calibration.

During operational use, the utilities, in combination with the measured attribute levels, allow computation of a Multi-Attribute Evaluation (MAE) value for each incoming message. The selector mechanism routes those messages to the recipient, prioritizing them with regard to MAE value, so that he or she always looks first at the most desired information. The selector can also use the MAE value to improve an individual's information-gathering efficiency. For example, most users select more information than is actually necessary to reach a decision. In this case, the model-based selector can reduce the transmitted message set by eliminating those messages which contribute less than some criterion value of MAE value.

1.2.3 System Development and TNG Demonstration. The methodology used to implement this system concept is described in Chapter 2. As a first step forward in developing AIS system software compatible with the TCO, the adaptive information selection model was applied at Perceptronics to the Tactical and Negotiations Game (TNG) simulation, which was modified for this specific purpose. The TNG requires its players to process information messages in order to make assessments concerning the military, intelligence,

economic, and negotiation activity of a small underdeveloped nation plagued by an internal revolution. This flexible yet controllable scenario provides multi-dimensional information for multi-faceted C3 decision making, and it has been demonstrated in past research to both maintain high face validity and produce reliable test results.

One of the difficulties faced by previous research users of the TNG has been the lack of a ground truth base for evaluating the quality of the player's decision performance. To overcome this problem, Perceptronics -- as part of a prior work effort -- performed a content analysis on written decision protocols generated in response to a large set of fixed messages during game-playing sessions by many groups of subjects. The output of this analysis was a set of four plausible specific states of the world (i.e., enemy strategies) for each situational content area (military, intelligence, economic, negotiation), with one alternative in each case clearly indicated by a consensus of player opinion as the correct one, i.e., the "school solution." This ground-truth base, together with the computerization of the TNG, has greatly increased its general usability as a research tool, especially for the study of complex C3-related decision processes.

The efficacy of the AIS system in message distribution and prioritization was demonstrated via a 'gauge-test' in the TNG simulation environment. The details of the in-house simulation and preliminary system evaluation are given in Chapter 3. A user's guide to the in-house system simulation is provided in Chapter 4. Functional flow-charts and source code listing for the training and testing phases are available in Appendices A and B, respectively.

1.3 AIS Transfer to TCO Simulation

1.3.1 TCO. The intent of the one-year effort described here has been to design and develop an AIS system that can, with minor modifications, be

transferred to the TCO simulation on the MTACCS Interim Test Facility. TCO will be one of eight functionally oriented tactical systems included in the MTACCS concept, slated for operation in the 1980's.

TCO will support commanders and their staff in carrying out their functions in the areas of operations and intelligence, including planning, intelligence production, and the monitoring and directing of tactical operations. The system will provide this support to ground elements, air elements, and Marine Air-Ground Task Force (MAGTF) Head-quarters.

TCO functions can be grouped into five top-level areas, these are:

(1) Operational Support
(2) Intelligence Support
(3) Fire and Air Support
(4) Logistics Support
(5) Personnel Support

TCO will perform these functions by offering commanders prompt, timely and accurate information for their consideration.

1.3.2 MTACCS Interim Test Facility. Alternative approaches to the automation of MTACC systems will be tested on the minicomputer-hosted MTACCS Interim Test Facility. The MTACCS Interim Test Facility provides for:

(1) The capability to support the evaluations of automation concepts for TCO and other MTACC systems.
(2) The representation of the tactical environment in which the MTACC systems are expected to function.
(3) Control of this representation by means of graphic and alpha-numeric terminals manned by Marine Corps personnel acting in an interactive man-machine mode.

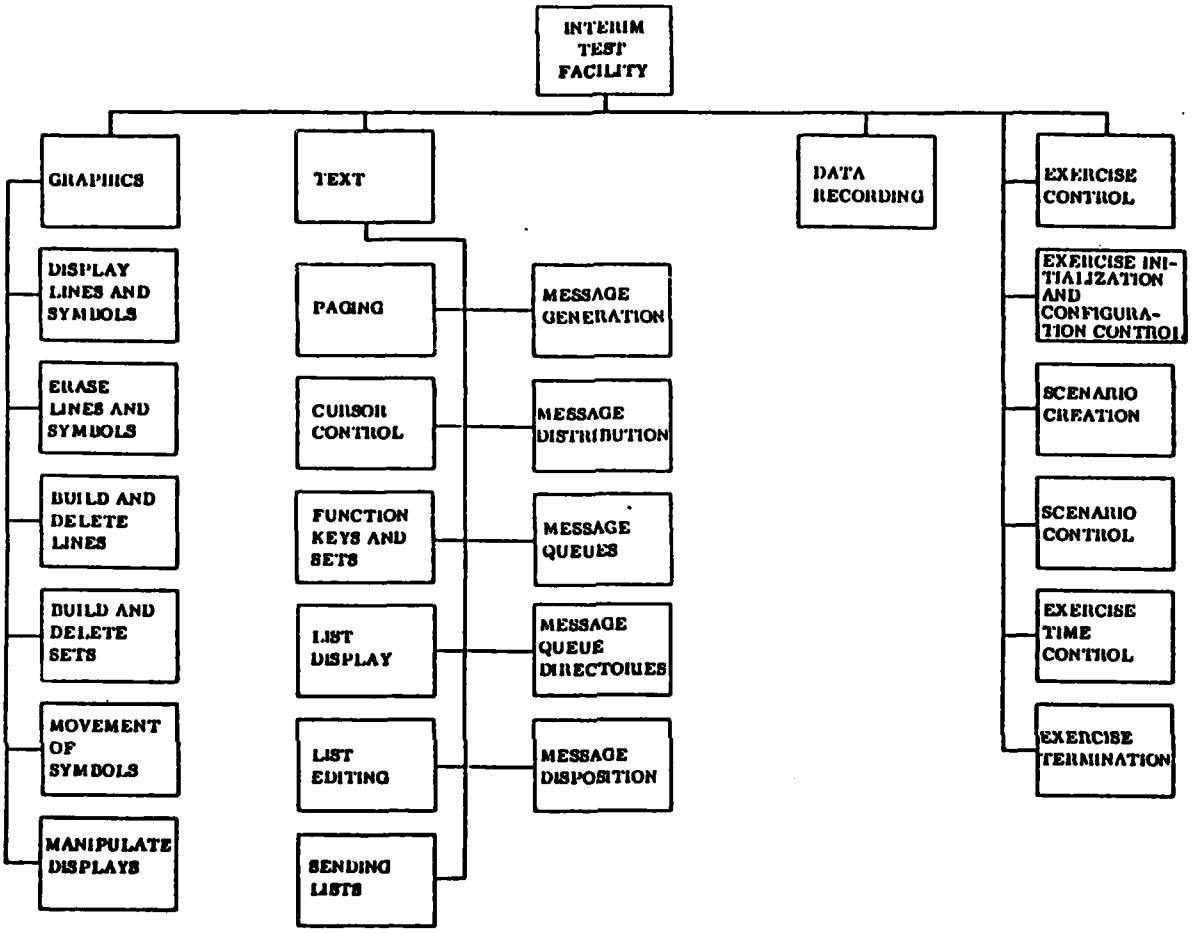


FIGURE 1-2. TEST FACILITY FUNCTIONS

The MTACCS Interim Test Facility can be considered an operationally-oriented laboratory where requirements are defined, tested, refined, and analyzed before they are implemented on the battlefield. As such, it is considered an ideal test facility for evaluation of the proposed AIS system. The software functions performed on the MTACCS Interim Test Facility are included in Figure 1-2.

1.3.3 Exercise Scenario. The exercises performed on the Interim Test Facility center on a scenario, i.e., a sequence of events and data inputs that emulate an actual military operation. Current tests are based on a Marine Amphibious Force (MAF), as the friendly force, making an amphibious landing across the beach at Camp Pendleton. The actual scenario represents a portion of operations ashore. The aggressor or enemy force is tailored after likely adversaries. All information pertaining to the aggressor is obtained from the U.S. Army aggressor's handbook. The basis for definition of friendly forces is the Landing Force Operational System Study (LFOSS), a document that provides a description of friendly forces in the future.

1.3.4 Transfer Requirements. A review of the messages resident in the database of the TCO simulation reveals that the attributes required by the current AIS software are all available or derivable from message headers. If the message header changes, only a small redesign of the AIS system would be necessary, due to its modular design. Test facility personnel are currently engaged in bringing up a version of the C language compiler to support their simulation. Since the AIS is written in the C language, it is expected that no significant software modifications will be required to make the AIS software compatible with the MCTSSA Interim Test Facility operating system.

Two different implementation concepts will be examined at the time of AIS software transfer to the MTACCS Interim Test Facility. These are discussed in Section 5 of this report. AIS system evaluation at the MTACCS Interim

Test Facility is planned to take two forms: (1) comparative testing with the existing manual routing channel, and (2) subjective evaluation of system effectiveness. These concepts are discussed in greater detail in Section 5.

2. ADAPTIVE INFORMATION MANAGEMENT

2.1 General

The goal of the 12 month program was to demonstrate the feasibility of modifying Perceptronics' AIS methodology for the Marine TCO system application as simulated on the MTACCS Interim Test Facility. This was performed by designing and implementing an in-house system incorporating the AIS methodology. This system uses a scenario similar to one used at the Interim Test Facility. The AIS aids TCO operations by managing transfer of critical information among data sources, tactical data bases, and data users. In brief, the AIS functions by characterizing messages along a measurable set of attributes. The attributes comprise such factors as message content, area, age, familiarity, accuracy, and geographic locale. The model computes an aggregate Multi-Attribute Evaluation (MAE) of a message as a selection criterion. The information selection policies of different users are then modeled for each tactical situation as distinct vectors of attribute weights. The set of multi-attribute models inherent in the AIS serve the following functions:

(1) Automatically routing messages from sources to various users.

(2) Automatically reprioritizing the queue of messages as command situations change.

2.2 AIS Aiding Concept

Figure 2-1 illustrates the AIS aiding of the TCO information dissemination function during the conduct of an exercise on the MTACCS Interim Test Facility.

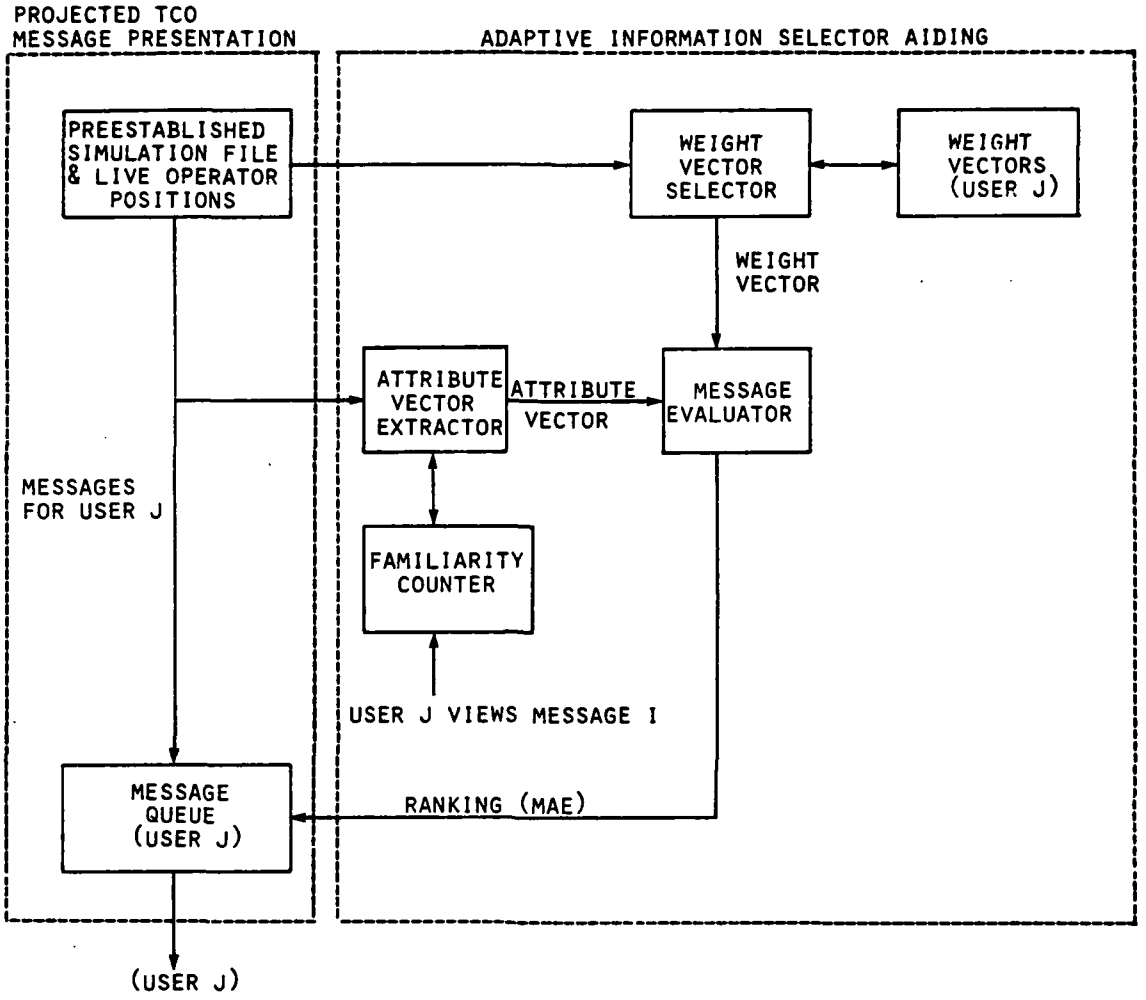


FIGURE 2-1. ADAPTIVE AIS AIDING OF SIMULATED TCO FUNCTIONS ON MTACCS INTERIM TEST FACILITY

During a typical TCO exercise on the MTACCS Interim Test Facility, the computer system serves the exercise players at tactical positions and a member of the Control and Simulation Team (CST) at control positions. The data processing system supports the entry and transmission of messages originating from both the pre-established simulation file and the live operator positions. Exercise operators assigned to each tactical position perform their normal military functions. They respond to stimuli from other operators, from CST members and from outputs of the simulation file. The system contains routing tables to control the distribution of messages. The routing tables are based on message type and originating node. They provide the preliminary distribution list for each message. The operators can add one recipient to the message distribution list without having to change the standard routing table. They possess temporary manual override capability on the standard routing table by designating a specific destination for a message at the time the message is sent. Further, for each user, separate message queues exist.

The Adaptive Information Selection will aid the projected TCO routing channel (in the performance of these exercises on the test facility) by automatically routing the messages and prioritizing the messages in each user's queue by utilizing the multi-attribute evaluation model approach. The specific functions of the various elements of the AIS as shown in Figure 2-1 are discussed below.

(1) Attribute Vector Extractor. The message attribute extractor "extracts" the attribute levels from each message (supplied by the pre-established simulation file and line operator positions) based on its header, content area, and tags. This information is partly contained in a table look-up format for each message ID.

(2) Familiarity Indicator. This is a counter for determining the number of times specific users have viewed a particular message. It assigns a value to the familiarity attribute level based on this count.

(3) Message Evaluator (ME). The ME function weights and aggregates the constituent attributes according to each specific user and command situation. Using a set of Multi-Attribute Evaluation (MAE) models, it performs the following functions:

(a) It determines a distribution list of recipients (user 1,.....in Figure 2-1) for each message which it recommends to the exercise operator (manual router). The exercise operator retains the prerogative to accept or reject this recommendation.

(b) It prioritizes or ranks the messages in each user's queue according to changes in the command situation as reflected by corresponding values in the weight vector provided by the WEIGHT VECTOR SELECTOR.

(4) Weight Vector Selector (WVS). The WVS function is to select appropriate weights per attribute for each user as the command situation changes.

2.3 System Organization

2.3.1 Attribute Definition. The additive attributes are those factors which act in a trade-off fashion (an increase in one factor will compensate for a loss in a second factor) and discriminate between user policies. The attributes are derived from the message headers and from the database. A representative set of attributes determined for the MTACCS simulation are:

Attributes A_1 to A_4: content area. Each message is categorized in the header according to its purpose. By analysis, this information can be partitioned into a set of 4-6 meaningfully distinct content areas for each user. These membership areas define the content

attributes. Presence of information in a given content area results in a unit level for that attribute, a zero level otherwise.

Attribute A_5: message age. The message header lists the time of origination of the message. The age is derived using the system clock.

Attribute A_6: specificity. The message could be summary or detailed. This attribute may be used to portray a message that was complete or incomplete.

Attribute A_7: familiarity. The number of times the user has seen the message. This is maintained in an internal table. This attribute may be simplified to whether the message was seen or unseen.

Attribute A_8: priority. This attribute portrays the importance of a message.

Attribute A_9: locale. This attribute refers to whether a message was generated locally or otherwise.

2.3.2 Attribute Weight Vector. Each user is assumed to have a policy of information selection that depends on his responsibilities and on the current command situation. This policy is reflected in a separate vector of attribute weights for each distinct situation. For example, an intelligence officer may have a high value for political information during a surveillance situation but less so during an attack. Of course, all combinations of user and command situations need not exhibit a distinct weight vector. Figure 2-2 shows how a given weight vector may be common to several different command situations. Minimizing the number of weight vectors in this fashion reduces the required training time.

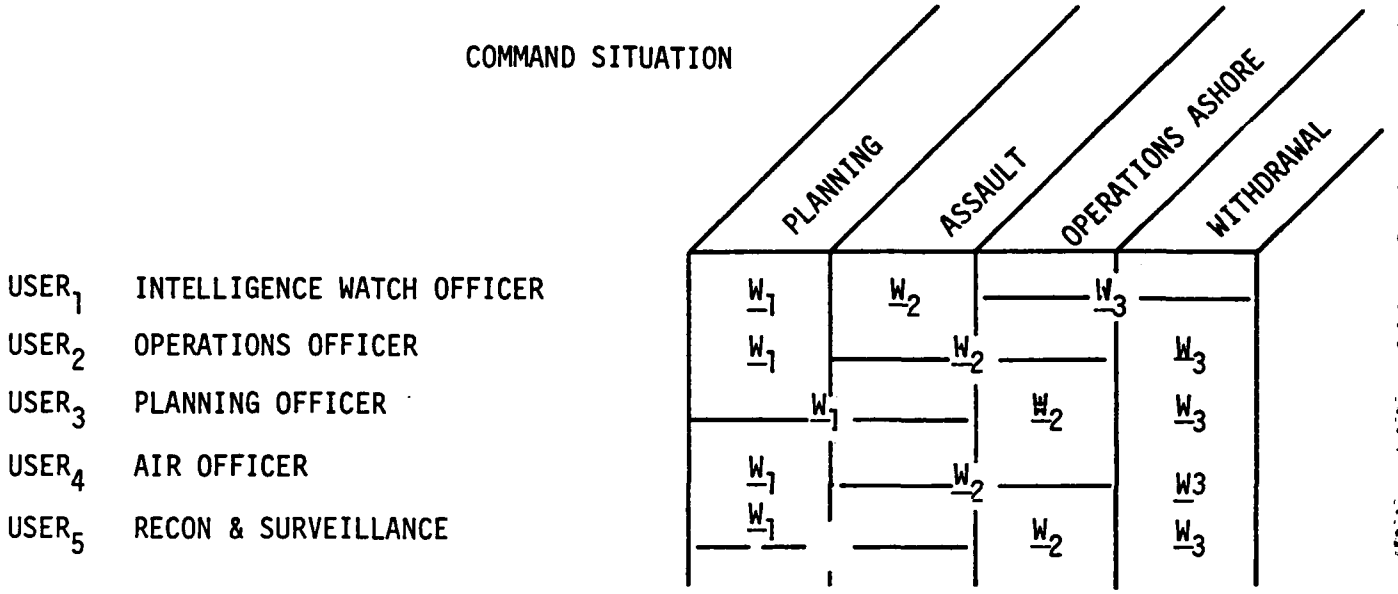


FIGURE 2-2. TCO WEIGHING KEYED TO COMMAND SITUATIONS

Estimates for a commander's attribute weights, or his utility for that attribute, are provided by the adaptive portion of the model. The weights are adjusted (or calibrated) during sessions where a commander performs the tasks required in the scenario by choosing freely among a menu of possible information items. The model begins with equal weights assigned to each attribute and then dynamically adjusts them in accordance with a simple training rule.

The dynamic utility estimation technique is based on a trainable, multi-category pattern classifier. Figure 2-3 illustrates the mechanism. As the user performs the task, the on-line utility estimator observes his choices among the available information categories or messages, and views his decision making as a process of classifying patterns of information attributes. The utility estimator attempts to classify the attribute patterns by means of a linear evaluation (discriminant) function. These classifications are compared with user's choices. Whenever they are incorrect, an adaptive error-correction training algorithm is used to adjust the utilities. This algorithm is described more fully in Section 3.1. A comprehensive discussion of this technique can be found in Freedy, et al (1976), or Steeb, et al (1977).

2.3.3 Training Algorithm. On each trial or training sequence, the model uses the previous evaluation weights (w_j) for each attribute (j) to compute the multi-attribute evaluations (MAE_i) for each available information category (i).

$$MAE_i = \sum_j w_j a_{ji}$$

The model predicts that the user must always prefer the available information item with the maximum MAE value. If the prediction is correct (i.e., the user chooses the information item with the highest MAE), no adjustments are made to the utility weights. However, if the user chooses a message having a MAE less

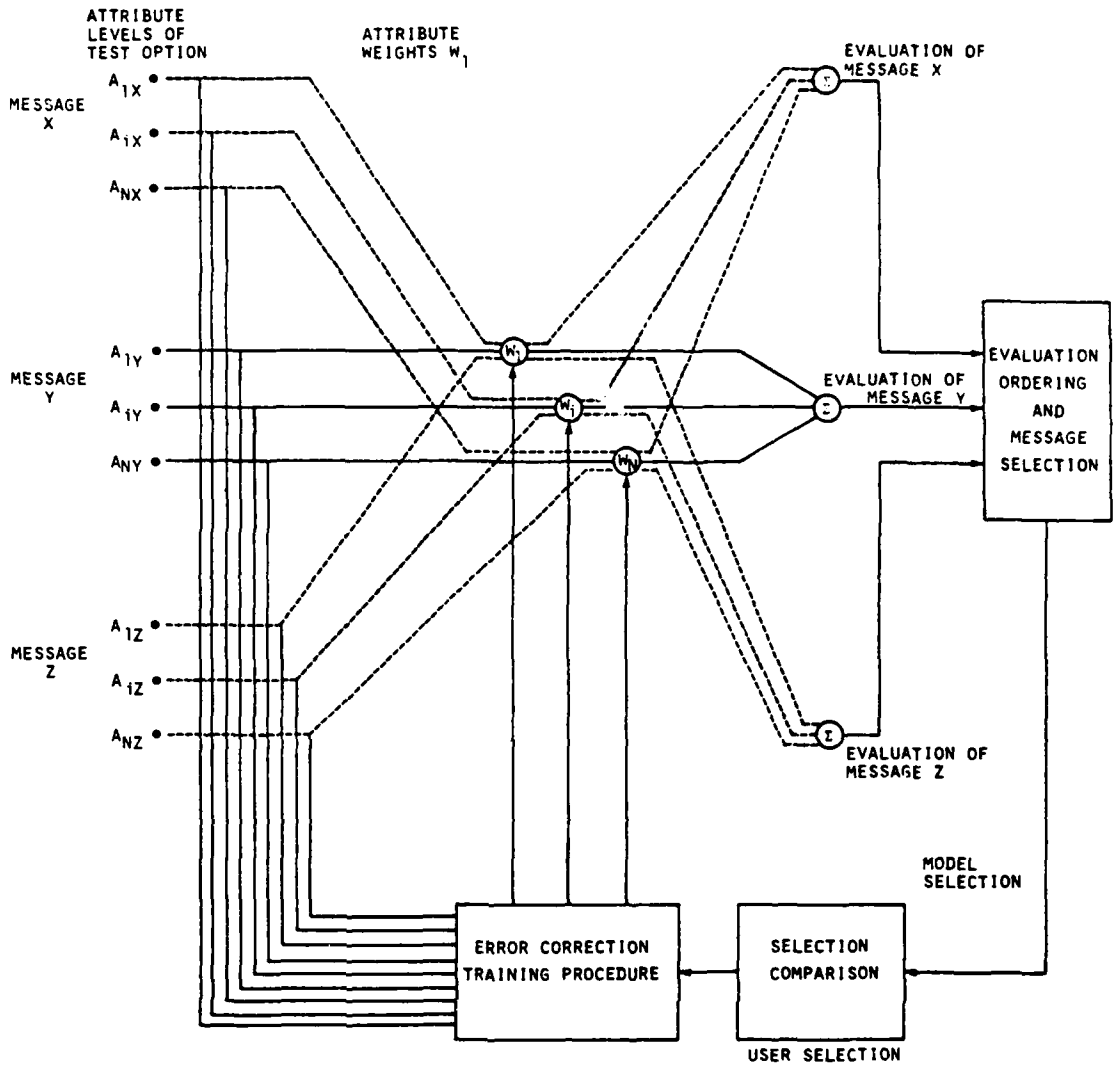


FIGURE 2-3. SCHEMATIC REPRESENTATION OF ADAPTIVE EVALUATION ESTIMATOR

than that of the predicted message, the model then adjusts the evaluation weights by pairing the chosen category with the predicted category and applying the error correction training algorithm.

The model uses the previous evaluation weight (w_j), for each attribute (j), and the difference of the attribute levels of the chosen message with the predicted message ($a_{jc} - a_{jp}$), to generate the new evaluation weight (w_j) for the $j^{\underline{th}}$ attribute. The amount of adjustment that takes place is governed by the constant λ.

$$w_j = w_i + \lambda (a_{jc} - a_{jp})$$

In this manner, the evaluator "tracks" the user's information selection behavior and learns his utilities or weights for information attributes.

3. IN-HOUSE SIMULATION SYSTEM

The Adaptive Information Selector simulation uses a scenario taken from an already existing C3 simulation called the Tactical and Negotiations Game (TNG). The TNG requires one or two players to process information messages and make decision assessments concerning the military, intelligence, economic and negotiation activity of a small underdeveloped nation plagued by an internal revolution. The AIS implementation uses a subset of the message file of the TNG. The messages that will be used contain six parameters from which the multi-attribute utilities will be determined. The current attributes used are: content area, age, detail, familiarity, priority and locale. The attributes indicate what a message is about, how old it is, whether it is a detailed or summary message, whether it has been seen yet, its priority and its origin. Figure 3-1 shows the attributes of two such messages. The simulation takes place in two phases. The first phase is the calibration phase where the Multi-Attribute Evaluation (MAE) weights of the model are adjusted while subjects evaluate messages at a terminal. The second phase is the verification phase where subjects rate and/or select messages which are being routed and prioritized at their terminals. The model weights from the calibration phase are used during the testing phase to calculate the MAE values used for routing and prioritizing of messages. A detailed discussion of each phase including documentation of the principal functions making up the software of each phase is given below.

3.1 Calibration Phase Overview

The calibration phase involves randomly generating message attribute values for pairs of messages. Each pair is presented to either a real or simulated subject. The subject is then required to select the message which he deems more important to him based on the message attributes and the current command situation. The MAE model predicts user choice based on its currently established parameters. The MAE model parameters are adjusted only if the model

COMMAND SITUATION: PRE-MOBILIZATION

Message	Content Area	Age	Specificity	Familiarity	Priority	Locale
A	negotiations	recent	detailed	unseen	high	adjacent
B	economics	old	detailed	seen	low	adjacent

Select A or B

FIGURE 3-1

fails to correctly predict the user's response. The adjustment rule is illustrated in Table 3-1. In the case of a simulated subject, the choices are guided by the attribute weights used to model the behavior of a real subject.

The MAE model parameters are calibrated under four separate command situations. Calibration of a new command situation is initiated only when the model weights for the previous command situation are 'stabilized'. A particular command situation is considered 'stabilized' if a test of convergence is obtained. Convergence is determined by one of two possible tests (see Section 4 for details of tests). Each test uses a sliding window over which convergence is evaluated. The training phase is concluded when the MAE model parameters for each of the four command situations have converged.

3.2 Testing Phase Overview

The testing phase of the Adaptive Information Selector simulation incorporates the weights from the calibration phase and computes the Multi-Attribute Evaluation (MAE) values for routing and prioritizing of messages based on these weights. Routing consists of messages from the message file being placed, at preset intervals of time, in one of a number of subject's bins (currently two). The subject bin selected for the message is determined by either the calculated MAE value or by a random number generator, depending on the particular control case the system is executing.

The efficacy of the training phase (i.e., how well the trained weights match the subject's role or policy) is determined during the testing phase. The testing is performed using one of two possible designs or approaches. In the first approach, prioritizing of messages is performed by having the system display to the subject his current bin of messages along with the message the the system has selected as being the most suitable. The most suitable message is based on the MAE computation using the trained weights. These take into

TABLE 3-1

WEIGHT-TRAINING RULE

Adjusted Weight		Previous Weight		Adjustment Factor*		CORRECTION DIFFERENCE: Chosen Information-Source Attribute Level		CORRECTION DIFFERENCE: Predicted Information-Source Attribute Level
$\hat{W}_1$	=	W_1	+	λ	·	$(A_{1c}$	-	$A_{1p})$
.		.		.		.		.
.		.		.		.		.
.		.		.		.		.
$\hat{W}_j$	=	W_j	+	λ	·	$(A_{jc}$	-	$A_{jp})$
.		.		.		.		.
.		.		.		.		.
.		.		.		.		.
$\hat{W}_N$	=	W_N	+	λ	·	$(A_{Nc}$	-	$A_{Np})$

* λ is a constant which influences the rate of training.

account specific command situations for a given subject. The subject then rates (on a scale from 1 to 5) the system recommended message. This rating indicates the subject's view on how suitable the system selected message was in terms of his prevailing policy. Statistics are kept on each subject's accumulated ratings of the system selected messages. In the second approach, the subject himself can select a message from the queue in his bin. The system compares his selection with its selection based on the MAE computation by calculating statistics that provide a measure of closeness between the subject's and the system's selection of messages. Either or both of these two design options should accurately test the validity of the trained weights.

To a limited extent, the efficacy of the MAE-based routing and prioritization function is established by the use of control cases. Each subject is tested using three separate control cases through which the system cycles. The first control case employs 'random' routing and 'random' prioritization of messages. During the second control case, the system performs random routing and MAE based prioritization. During the third control case, the MAE computations are used for both routing and prioritizing of messages. Using this evaluation scheme, a measure of the advantage of MAE based routing and prioritizing over purely random message presentations can be established. If the MAE model is effective, the third control case should produce significantly higher ratings than the first over a reasonable time duration. The ratings for the second control case should fall somewhere between those produced for the first and third control cases. In the Test Facility, however, MAE-based routing and prioritization efficacy will be measured against the existing 'hard wired' distribution list. This will provide a more realistic measure of the advantage of MAE-based routing and prioritization functions.

As described earlier, the scenario used for the in-house simulation is an adaptation of a command and control simulation system called the Tactical and Negotiations Game (TNG). Currently, there are sixty messages making up the

scenario file. At present, two subjects can receive messages at their respective stations (terminals). This number can easily be extended to serve additional subjects or stations with slight modifications to the software. At the end of the testing phase simulation, the cumulative statistics printout for each control case and command situation is presented.

3.3 Testing Phase Design

In the testing phase, each subject is seated at a terminal and apprised of a specific responsibility or given a specific policy role to follow. Typical user responsibility roles assigned to him are that of an intelligence officer or a logistics officer in a dynamic tactical environment. A typical view of the information displayed to the user at his terminal is shown in Figure 3-2. Each subject's display basically consists of a list of all the non-viewed messages which the system has sent to his station. Two possible message-related commands can be entered by the subject. These commands, along with a brief description of each, are listed in Table 3-2.

Each system user has a message bin that stores messages routed to his station. The display control system maintains a dynamic display for each user. As messages are routed, they are automatically displayed on the user's terminal. The resulting message queue held in the subject's bin is displayed strictly in terms of message attributes. The dynamic nature of the bins allows the message display to be updated, and messages to be deleted after viewing. Currently, available display space allows no more than 10 messages to be stored in a bin at any instant. When ten messages have accumulated, the display system uses a first in - first out (FIFO) philosophy to display the the most recent messages.

The possible commands that can be issued by a user are; 'N', 'R', and 'S'. The command 'N' tells the system to select the next appropriate message for

MESSAGE ATTRIBUTES CS: COMBAT

MESSAGE	CONTENT AREA	AGE	SPECIFICITY	FAMILIARITY	PRIORITY	LOCALE
0	economic	old	summary	unseeen	low	own
1	intelligence	immediate	summary	seen	high	adjacent
2	economic	recent	detailed	seen	low	own
3	military	immediate	detailed	unseen	high	own
4	military	old	summary	seen	low	own
5	negotiations	recent	detailed	unseen	low	adjacent
6	intelligence	old	summary	unseen	high	own

COMMAND: Enter (N or S)?

3	military	immediate	detailed	unseen	high	own

THE COMMANDER OF THE 102ND DIVISION REPORTS THAT HIS UNITS ARE CONTINUING TO SEARCH FOR ENEMY TROOPS AND SUPPLIES. ONE BRIGADE HAS FOUND A TUNNEL SYSTEM CONTAINING AMMUNITION AND FOOD SUPPLIES BUT HAS SIGHTED ONLY A FEW ENEMY TROOPS WHO WERE APPARENTLY RETREATING.

FIGURE 3-2. MESSAGE DISPLAY IN TESTING PHASE SHOWING USER BIN AND DISPLAYED MESSAGE

TABLE 3-2

USER COMMAND SET

N	Command for MAE model to select a message
Rn*	User rating of the MAE selected message
Sn†	User command for message selection

* n is an integer in the range 1 to 5, used to rate the MAE selected message.

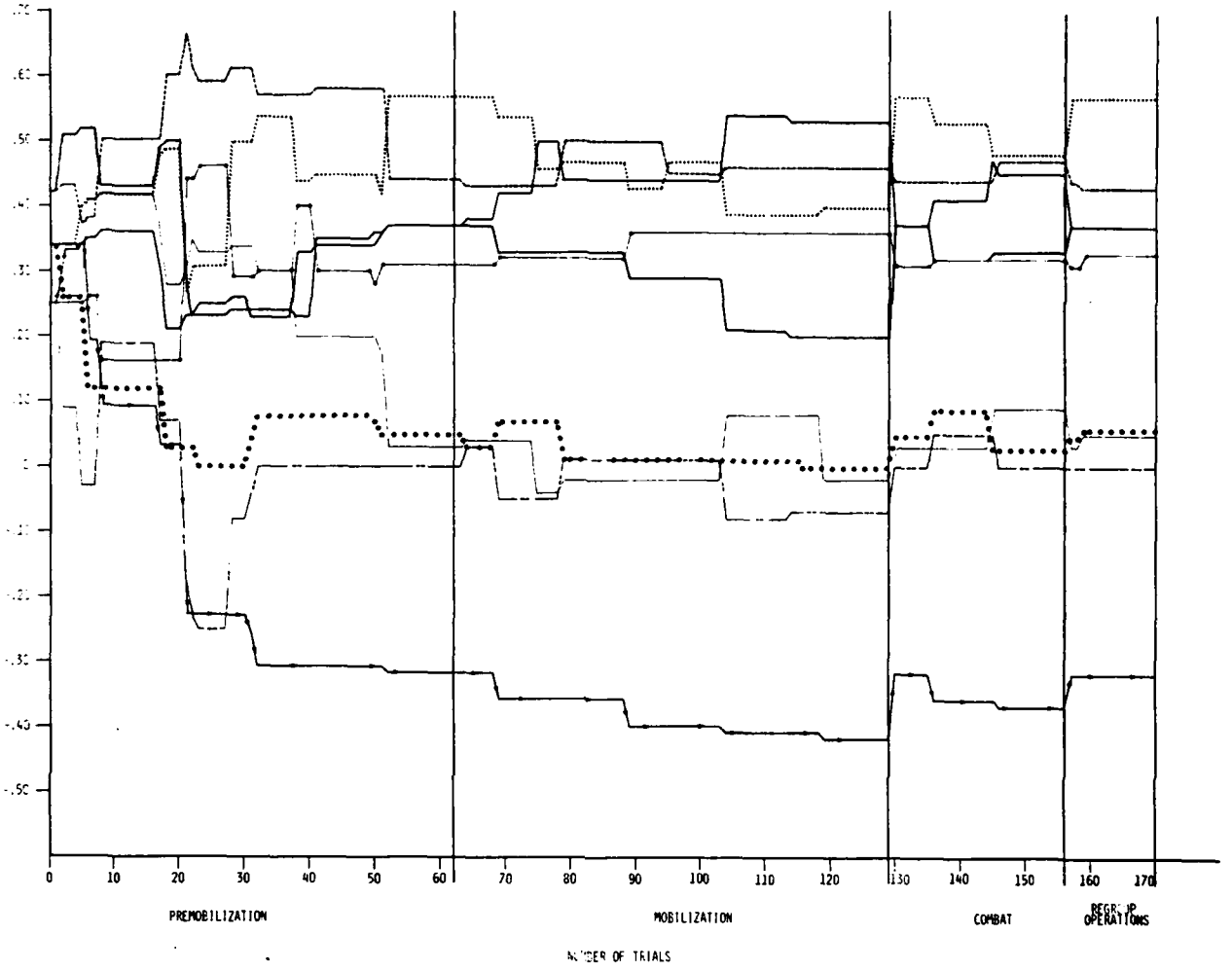
† n is an integer in the range 0 ~ 9. It is used to select which message from the user's bin is desired for display.

the user from those in his bin. This message and its associated attributes are displayed at the bottom of the screen as shown in Figure 3-2. The other messages sent to him until that point are displayed in terms of their attributes only. After viewing the system selected message, the subject enters a rating command which consists of an 'R' followed by a number from 1 to 5. This command allows the subject to rate what he thinks the value of the system selected message is to him, based on his assigned policy role.

The third command 'S', is followed by a digit from 0 to 9 which corresponds to the message in the subject's bin that he wishes to select. This message is presumed to be the one the model should have chosen given an 'N' command. The difference between this message and the system selected message is monitored, and statistics on system performance gathered.

The testing phase entails three separate control cases and four different command situations. Each of the four command situations is repeated for the three control cases. The three control cases are numbered 0, 1 and 2. The four command situations are *Planning*, *Assault*, *Operations Ashore*, and *Withdrawal*. Each command situation corresponds to a sub-scenario which is characterized by specific operations. Planning is characterized by a lull in combat with very little activity. In Assault, combat-ready operations are in process. In Operations Ashore, an actual battle is in process. Withdrawal is the period immediately following a combat situation. Subject responsibilities and, in turn, information seeking behavior, will vary according to the prevailing command situations.

A representative example of convergence behavior of the AIS simulation is shown in Figure 3-3. Four mission phases are present, as shown by the four segments of the figure. At the beginning of the session, all nine attribute weights are arbitrarily set to the same normalized value, .33. The weights quickly diverge from this value until, by the twentieth decision, they are



Military ——— Econimic ●——● Familiarity →→→

Intelligence ····· Age — - - — Priority ———

Negotiations - - - - - - - Specificity ●●●●● Locale ———

Figure 3-3

consistently ranked. Calibration continues until a convergence criterion is met (10 decisions without adjustment). At this point, the mission phase changes. The previously estimated weights are used as the starting values, and calibration continues until convergence is reached for this phase. The adjustment period is reduced (particularly in mobilization and combat operations) since less adjustment is required from the initial vector. Also, the number of trials to convergence could be further reduced if the message were chosen from a large number of options in each decision, rather than the pairwise choice made here.

Further experimental tests of the AIS will be performed at the MTACCS Interim Test Facility. A description of these tests is given in Section 5.

4. USERS GUIDE TO AIS SIMULATION

4.1 General

The AIS model is written in the C programming language and is run under the UNIX operating system. The simulation system consists of two separate programs which access a shared data base. The first program performs the initialization of the system by calibrating a weight vector for a given user. The second program provides for verification by allowing the user to monitor the model in prioritizing messages using the previously trained weight vector.

Each program has parameters that control program execution. These parameters reside in an external file. When parameter changes are desired, the new parameter values may be entered into these files and linked to the system. A list of modifiable parameters in each program is given in Table 4-1.

4.2 Model Calibration

In this phase, a user is assigned a specific policy or role to follow. The system displays two messages at a time to the user. The user's assignment is to select that message which he feels is more important to him under his designated role. The model adjusts its weights in accordance with the user's prevailing policy under each of four command situations. The convergence of the weight vector is determined by using one of two tests. The test that is employed is determined by the user when the system is loaded. The algorithm used by each of the two convergence tests is given below.

When user weights under all of the command situations have converged, the program prints: (a) the resultant statistics (computed progressively) and (b) the final weight vector for the user. Appendix A provides the functional flowsharts and source code of the major modules that make up the training phase.

TABLE 4-1

MODIFIABLE PROGRAM PARAMETERS

MODEL CALIBRATION PHASE

DESCRIPTION		PARAMETER
Simulated Subject Weights	-	simwt [NSIT][NATT]
Initial Subject Weights	-	weight [NSIT][NATT]
Real Subject Flag	-	simul
Convergence Window Thresholds	-	thold1, thold2
Convergence Method Flag	-	method

TESTING PHASE

DESCRIPTION		PARAMETER
Subject Weights	-	weight [SUBJ][NSIT][NATT]
Message Inter-Arrival Time	-	vect [MSG]
Command Situation Duration	-	tcs [CSI]
Number of User Terminals	-	NSUB

Convergence Tests

Two convergence tests are available in procedure 'test'. Both use a convergence window and threshold as specified in procedure 'data'. The two methods are described below:

Method 1

Behavior over a window, w, is recreated and evaluated with the new weights. Convergence is declared if the current weights predict the behavior over the windown sufficiently well. The algorithm is as follows:

If (SUM<thold1) then convergence declared
otherwise convergence not declared

where:

$$SUM = \sum_{i=t-w+1} \alpha_i \left| \underline{\Delta \ell}_i \cdot \underline{W}_t \right| \qquad \sum \alpha_i \left| \underline{\Delta \ell}_i \cdot \underline{W}_t \right|$$

where:

- t = current trial
- i = index of trials
- w = window
- $\underline{\Delta \ell}_i$ = attribute level difference ($\underline{\ell}_A - \underline{\ell}_B$) vector for trial i
- $\underline{\ell}_A$ = attribute level for message A; $\underline{\ell}_B$ = attribute level for message B
- $\underline{W}_t$ = current weight vector
- α_i = 1, if current weights don't predict the actual choice
  -0, otherwise

Method 2

In this method the amount of adjustment of the model is accumulated over the convergence window. The amount of adjustment is a function of the weights as they existed at each trial rather than the current weights. The algorithm is as follows:

If (SUM<thold2) then convergence declared
otherwise convergence not declared

where:

$$SUM = \sum_{i=t-w+1}^{t} \text{adjusted}(i) = \sum_{i=t-w+1}^{t} |\lambda(i)|$$

where:

w = number of adjustments per window
adjusted (i) = absolute value of the amount of weight adjustment on trial i as returned by procedure 'adjust'

4.3 Testing Phase

Currently, a minimum of three terminals are required to run the simulation program for the testing phase. The capability for using more than two subjects is easily achieved with the addition of extra terminals. One terminal is used as a master terminal and controls the simulation; the addition terminals play the role of user stations. The user stations are standard CRT terminals.

All terminals must be turned on and logged in under one user identification

(i.e., Dave). Before beginning the simulation, the program disables the CRT terminals being used, and enables them again on exit.

To start the simulation, the operator simply enters: "AIS x y" followed by entering the carriage return key. The x, y pair refer to the terminals that will be used. A message describing the users responsibility and possible actions appears on the terminal. When all users are ready, the system starts routing messages to each user. The statistics, which include the mean rating and the variance of the ratings for each subject, are printed on the master terminal after the termination of each control case and command situation.

In addition to the user commands for the AIS model, two system commands are available for controlling the AIS model. The time that each subject spends in each command situation can be controlled both by the system and the subject. If the subject wishes to proceed to the next command situation, he simply enters the command '?'. The subject may also temporarily halt the simulation and restart if at a later time by entering the command 'h'. A list of the possible system commands is given in Table 4-2. The functional flowcharts and a source listing of the major modules of this phase are included in Appendix B.

TABLE 4-2

TEST SIMULATION CONTROL COMMANDS

COMMAND	MNEMONIC		USAGE
N	'next'	-	display next msg, as chosen by MAE
Rn	'rate'	-	used to supply a users rating of n to MAE chosen msg.
Sx	'select'	-	display msg x, user desires to view msg explicitly
H	'halt'	-	halt simulation temporarily
Pxy	'proceed'	-	proceed to control case x, command situation y. Default values are current control case and command situation.

5. AIS TRANSFER TO THE MTACCS INTERIM TEST FACILITY

5.1 General

The MTACCS Interim Test Facility at Camp Pendleton, California will be used to test and evaluate the Tactical Combat Operation System (TCO) design. Due to differences in the structure and environment of the test facility, some of the principal AIS functions will undergo slight modification prior to their installation. The software of the simulation development system is written in the C programming language under the UNIX operating system. Personnel of the test facility are presently engaged in bringing up a version of the C compiler. Only slight software modification is anticipated to make the current programs compatible with the Interim Test Facility operating system.

5.2 AIS Implementation

At present, two alternatives exist for implementation of the AIS model at the test facility. One alternative calls for imbedding the major software modules directly into the test facility software. The second alternative is to have the AIS model serve as a stand alone program, with the MTACCS Test Facility used as a backdrop or working environment. A final decision will be made after further examination of the benefits afforded by each alternative.

The remaining effort in implementing the AIS model will be in interfacing to the existing software of the test facility. Personnel at Camp Pendleton concur with us that the most expedient approach will be to perform the interfacing at the Interim Test Facility (ITF). This approach allows the remaining effort to concentrate directly on the MCTSSA application.

5.3 AIS Calibration

AIS calibration takes place in response to two types of behavior: (1) routing of messages to the various nodes, and (2) viewing or discarding of messages by the users. These two functions are illustrated in Figure 5-1. The first, routing, is shown in isolation in Figure 5-2. Here, a sequence of messages is generated from the scenario file, each with a distribution list of nodes to which it will be routed. The messages will be treated two at a time by the training algorithm. For each node, the messages will be evaluated according to the node's weighting policy. If a discrepancy between the predicted routing and the actual routing occurs, an error correcting adjustment will be made. For example, node 1 is scheduled to receive message 2 but not message 3. If an evaluation of the two messages using node 1's weights does not correctly predict this advantage, a correction of the weight is necessary. In abbreviated form, the adjustment is as follows:

$$[W_{11}, W_{12},\ldots W_{1N}]^{new} = [W_{11}, W_{12},\ldots W_{1N}]^{previous} - \lambda \{[A_{21},\ldots A_{2N}] - [A_{31},\ldots A_{3N}]\}$$

Continuing in the example, node 2 is sent message 2 but not message 1, and node 3 is sent message 3 but not message 2. Adjustments are made to the node 2 and 3 weight vectors if the predictions are correct. In this way, model predictions for each node are evaluated for each pair of messages, and systematic corrections are made for errors.

The second form of behavior, i.e., user responses to the messages received, results in a similar process of observation and adjustment. As shown in Figure 5-1, a given node will receive a sequence of messages. The operator will either view a given message or hold it for later viewing or discard it without viewing. The viewing is essentially an acceptance of the routing, while the holding is a rejection of the routing. Each pair of messages coming

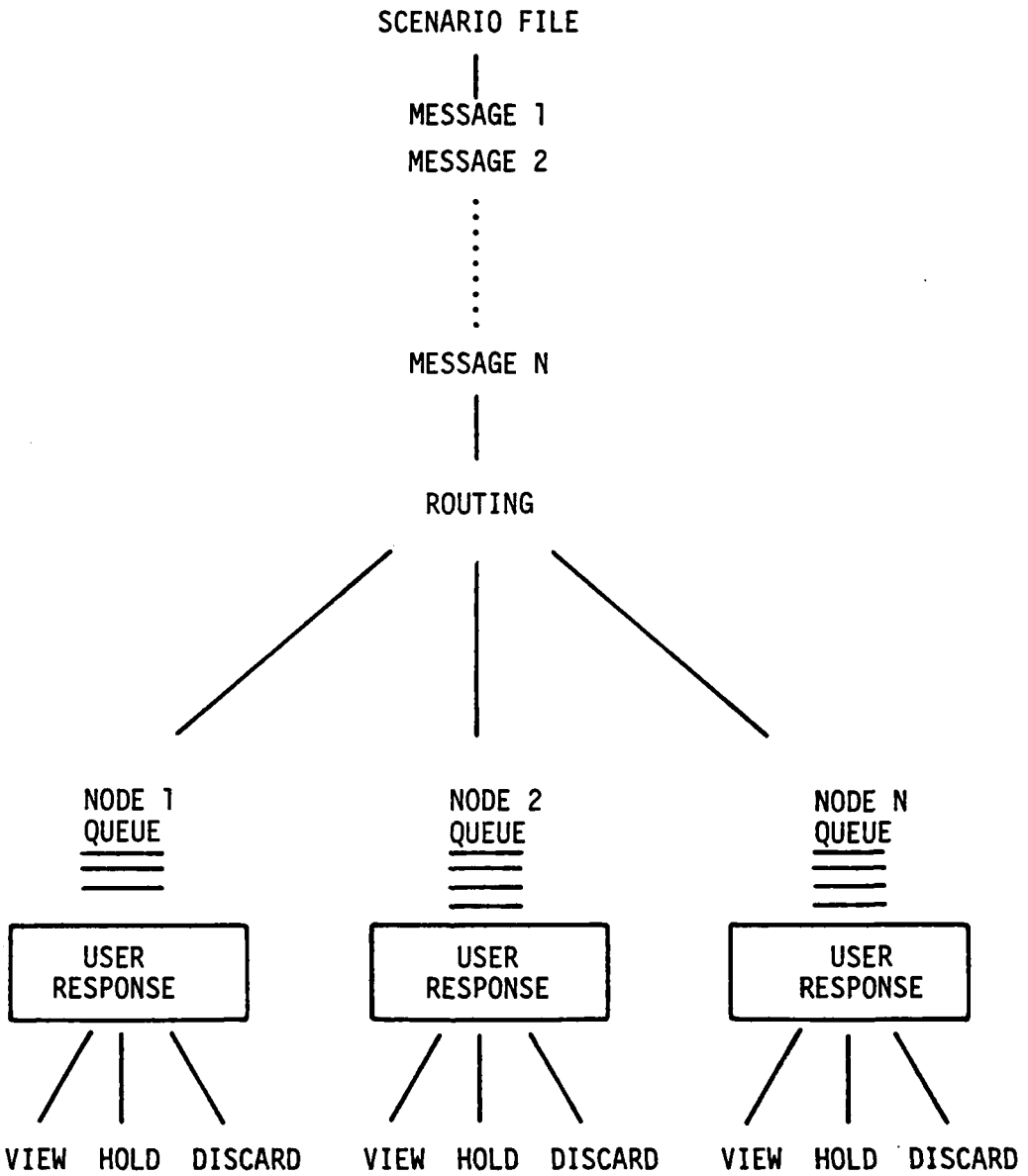


FIGURE 5-1. MESSAGE HANDLING BEHAVIORS IN INTERIM TEST FACILITY

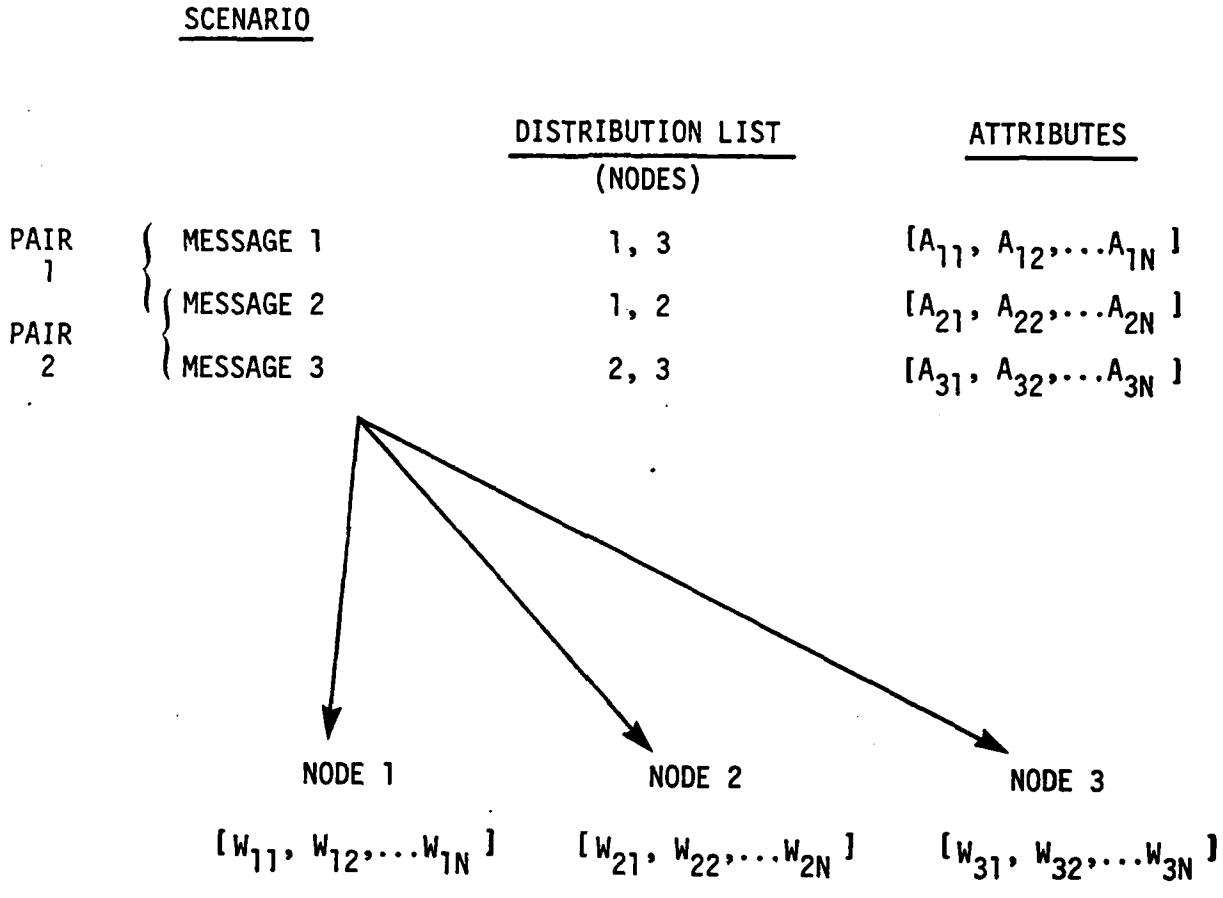


FIGURE 5-2. ROUTING BEHAVIOR

to a user is evaluated by the model and a prediction made as to the relative importance. If the user discards a message of predicted high importance, and holds one of low importance, an error correcting adjustment is made as before.

These types of adjustment, according to routing and according to response, are made in concert. Each adjusts the same sets of user policy weights. The use of two sources of behavior speeds the adjustment process and provides inputs from both router and user as to distribution policy.

5.4 AIS Test and Evaluation

AIS system evaluation at the MTACCS Test Facility is planned to take two forms: (1) side-by-side comparison with the existing manual routing, and (2) subjective evaluation of system effectiveness. In each case, information seeking policies will be estimated by the AIS for each recipient and command situation. Messages will then be routed to the various users. In the side-by-side comparison, the overall effectiveness of usage of the information will be determined for both the AIS-aided and the manual routing channels. The subjective evaluation will be more individually oriented. Each recipient will rate on an absolute basis the quality of the messages provided by the AIS process.

5.5 ITF Hardware/Software Capability

Hardware. The Interim Test Facility system is a minicomputer network comprised of:

1 DEC PDP 11/70 Minicomputer
8 DEC VT 52 Display Terminals
2 Vector General Graphic Displays

Software. The software available on the Interim Test Facility consists of:

(1) A Database Management System: DEC DBMS-11
(2) Time-Share Operating System: IAS Version II
(3) High Level Languages: COBAL, FORTRAN, MACRO, CORAL-66, and C.

5.6 Estimated Burden of AIS

The following is a breakdown of the costs of operating the AIS. These costs involve the additional memory needed for the software and the added response time due to the AIS. In the memory forecast (Table 5-1), total cost is reflected in the predicted number of line code needed to implement each principal function. The total number of lines is then translated into memory estimate in bytes. This does not include I/O routines or other system functions which should already be in memory. In addition, the necessary extra data areas, such as those required for the attribute level tags associated with each message, are given.

For the response time calculations, a worst case example is given. The example assumes a message pool of 105 messages with eight attributes each and three message recipients (users). The Router and Familiarity Indicator programs should add minimally to the response time of the AIS because much of the code involved in these routines already exists. Essentially, the only addition to response time is due to the Message Evaluator and Prioritizer. The calculations of execution time involved in these routines is shown in Table 5-2.

As of now, the worst case estimate by MCTSSA for message delay on the TCO is 5 seconds (based on message delay time observed on the former test facility). Our prediction of the AIS response time is about .05 seconds, which, compared to the existing response time, is negligible. Also, given the 32K partition of memory at the test facility, the 24K bytes of memory needed for the AIS will allow the model to perform in a stand-alone fashion.

TABLE 5.1 MEMORY FORECAST

Principal Function		Lines of Fortran Code
Familiarity Indicator		100
Weight Vector Selector		100
Message Evaluator		150
Prioritizer		100
Display Controller		150
	Total Lines	600
	Number of Bytes	20K
Data Areas	TOTAL MEMORY	4K
		24K

TABLE 5-2. AIS RESPONSE TIME (WORST CASE)

SAMPLE 105 messages
8 attributes
3 users

MAE CALCULATIONS:

105 x 8 x 3 x (1.0μ sec.) Floating Pt. add & multiply
= 8.8 m sec. + 10m sec. (overhead execution time)
= 18.8m sec.
To prioritize messages:
= 105^2 x 2.0μ sec. (for a compare & load)
= 22m sec. + 10m sec. (overhead execution time)
= 32m sec.
Total AIS response time = 51m sec.

Further discussion with personnel associated with TCO will determine the feasibility and desirability of integrating the AIS software into the exsiting system.

5.7 Preliminary Evaluation of the AIS Model Concepts

Before entering the final stage of the AIS model implementation and transfer to the Marine Test Facility, key Marine personnel associated with this project were invited to participate in a model demonstration. Two separate site visits and demonstrations were given which allowed LtC. Wickens, LtC. Malady, Maj. McDonough and Maj. Collins to participate in an AIS model simulation. On each visit, the model was first calibrated, and was subsequently followed by a model verification phase. The final results were encouraging, and the ensuing discussion led to the incorporation of additional features into the system.

The performance of the AIS model was closely monitored during the simulation, and computer printout gathered during each trial run verified the model to be performing as expected. The use of identical starting weights showed that the model was able to converge to different, yet similar, weights for each of the subjects. In the model verification phase, the efficacy of the AIS model to selectively route and prioritize messages according to user preferences was demonstrated. The objective rating of the model's ability to prioritize messages showed a very high degree of success, and was accompanied by a subjective rating in which the AIS mdoel was very highly rated by those involved in the demonstration. Further, the Marine Corps Personnel commented that the capabilities of the AIS model would be essential for effective future military operations.

The next step planned in the AIS model implementation involves the final decision on placement of the AIS model in the TCO environment. Primarily,

the decision is in regards to implementing the AIS model as a stand-alone unit with the TCO as a backdrop to AIS operations, or in modifying existing TCO software to accommodate the principal AIS functions.

6. REFERENCES

Interim MTACCS Test Facility. TCO Requirements Document, Marine Tactical Systems Support Activity (Camp Pendleton, CA), October 1977.

TCO System Description Document. Draft Report, Marine Tactical Systems Support Activity (Camp Pendleton, CA), December 1977.

Samet, M.G., Weltman, G., and Davis, K.B. Application of Adaptive Models to Information Selection in C3 Systems. Perceptronics (Woodland Hills, CA) Technical Report PTR-1033-76-12, Contract MDA903-76-C-241, Defense Advanced Research Projects Agency, December 1976.

DBMS-11 Data Base Administrator's Guide, Digital Equipment Corporation (Maynard, Massachusetts), DEC-11-ODABA-B-D, July 1977.

Freedy, A., Davis, K.B., Steeb. R., Samet, M.G., and Gardiner, P.C. Adaptive Computer Aiding in Dynamic Decision Processes: Methodology, Evaluation, and Application. Perceptronics, Inc. (Woodland Hills, CA), Final Technical Report PFTR-1016-76-8/30, Contract N00014-73-C-0286/NR 196-128, ARPA Order No. 2347, Code 455, Office of Naval Research, August 1976.

Steeb., R., Chen, K., and Freedy, A. Adaptive Estimation of Information Values in Continuous Decision Making and Control of Remotely Piloted Vehicles. Perceptronics Technical Report PATR-1037-77-8. (Woodland Hills, CA), August 1977.

Tactical Combat Operations System Detailed Requirements Document (Preliminary Material). Prepared by Computer Sciences Corporation for Marine Corps Tactical Systems Support Activity, September 29, 1978.

Interim MTACCS Test Facility System Operator's Manual. Prepared by Computer Sciences Corporation for Marine Corps Tactical Systems Support Activity, September 8, 1978.

Tactical Combat Operations System Software Requirements Evaluation. Prepared by Computer Sciences Corporation for Marine Corps Tactical System Support Activity, November 30, 1978.

TCO System Software Requirements Evaluation for Alternative II. Prepared by Computer Sciences Corporation for Marine Corps Tactical Systems Support Activity, December 15, 1978.

Keeney, R.L., and Raiffa, H. Decision with Multiple Objectives: Preferences and Value Tradeoffs, New York: Wiley 1976.

APPENDIX A

CALIBRATION PHASE

MODEL CALIBRATION PHASE

1. Introduction

The AIS model is implemented as a two-part program. Part one adjusts a weight vector for a user assigned a particular policy role. Part two provides validation of the calibration phase, including statistics on program execution. The model is provided with two convergence tests which are used in conjunction with the adjustment algorithm to adjust the weights of the users weight vector as the situation warrants.

2. Principle Functions

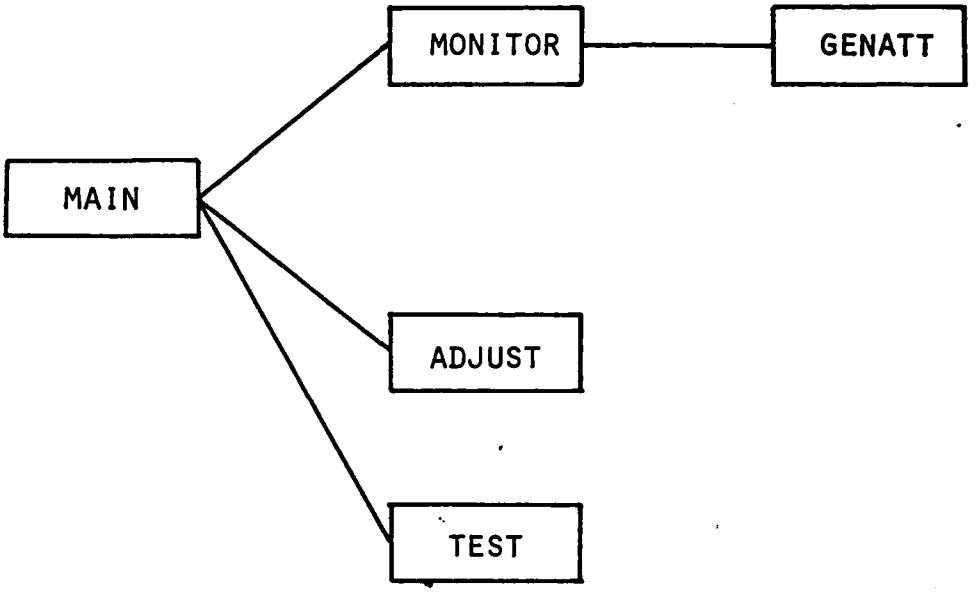
The training phase of the AIS model is broken into five major modules. They are:

(1)	MAIN	-	Main Program
(2)	MONITOR	-	Subject Monitor
(3)	TEST	-	Convergence Tests
(4)	ADJUST	-	Weight Adjustment Module
(5)	GENATT	-	New Message Attribute Selector

Figure A-1 shows the module invoking sequence of the five major modules.

2.1 Main

A functional flowchart for the module MAIN is given by Figure A-2. This module is responsible for overall program execution. The weight vector of a system user is given as a quadruple, with a distinct weight vector calibrated for each command situation. The system adjusts the weight vector of a given command situation until the weights satisfy a given convergence criteria. When the weights currently being adjusted have converged, the



MODEL INVOKING SEQUENCE OF CALIBRATION PHASE

FIGURE A.1

FUNCTIONAL FLOWCHART: TRAIN MAIN 1

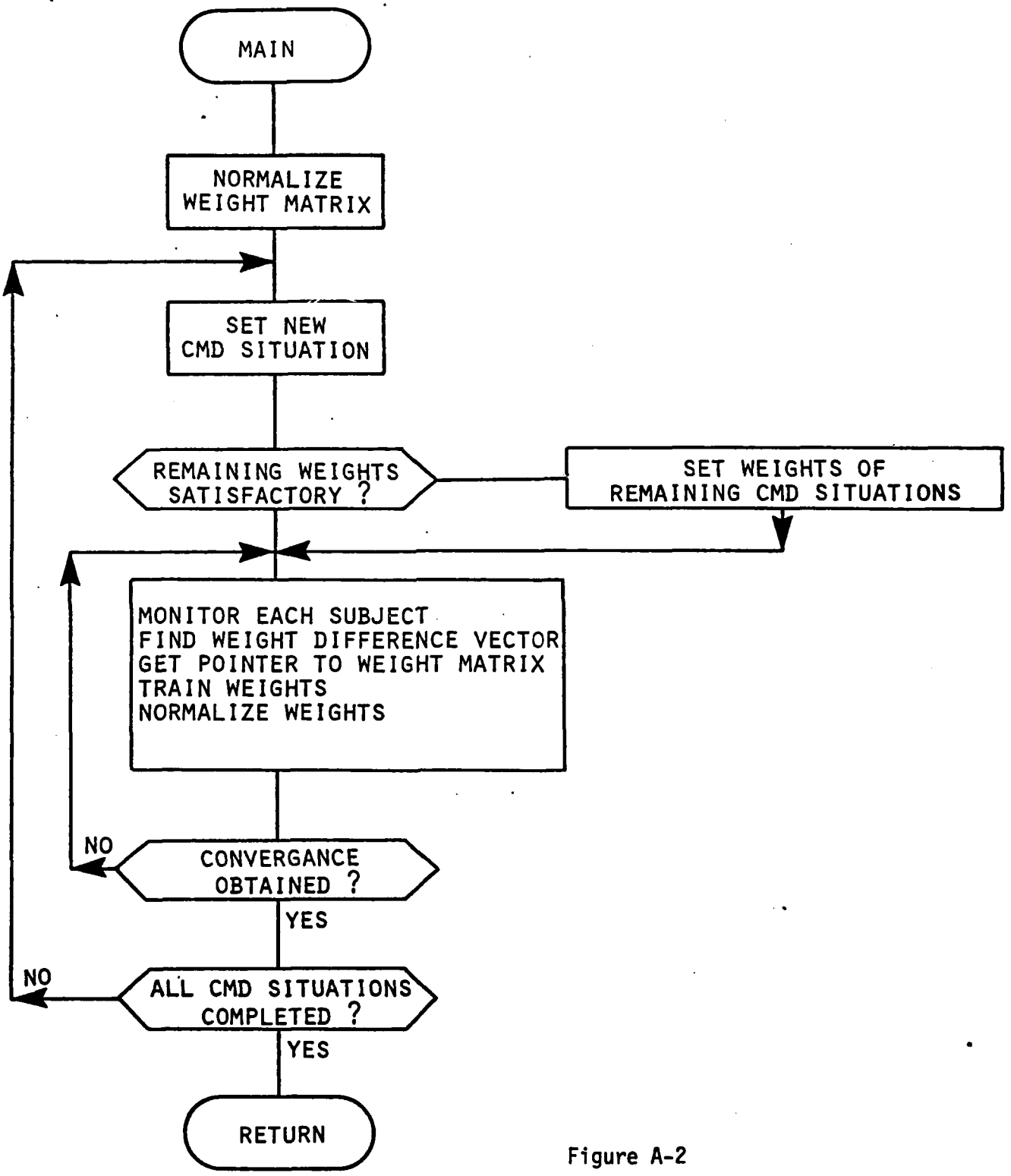


Figure A-2

weight vector for the next command situation is calibrated. When all command situations have been encountered and the corresponding weight vectors corrected, the calibration phase terminates.

2.2 Monitor

This module is responsible for selecting the next pair of messages, displaying them to the user and recording his response. The system is designed so that if a simulated subject is used, the model will make choices for the simulated subject based on an initialized weight vector. A functional flowchart for this module is given in A-3.

2.3 Test

The module that test convergence is given in functional form in Figure A-4. This is a function subroutine and returns the value 0 or 1. The value returned is zero if further adjustment is required; otherwise it is 1.

2.4 Adjust

When adjustment is required, this module is invoked. Three distinct forms of modification are possible. They are:

(1) Fixed increment rule
(2) Absolute-correction rule
(3) Fractional-correction rule

The particular type of correction that takes place is given as a parameter to this subprogram. A function flowchart describing this module is given in Figure A-5. The program returns the amount of correction which takes place to the calling program.

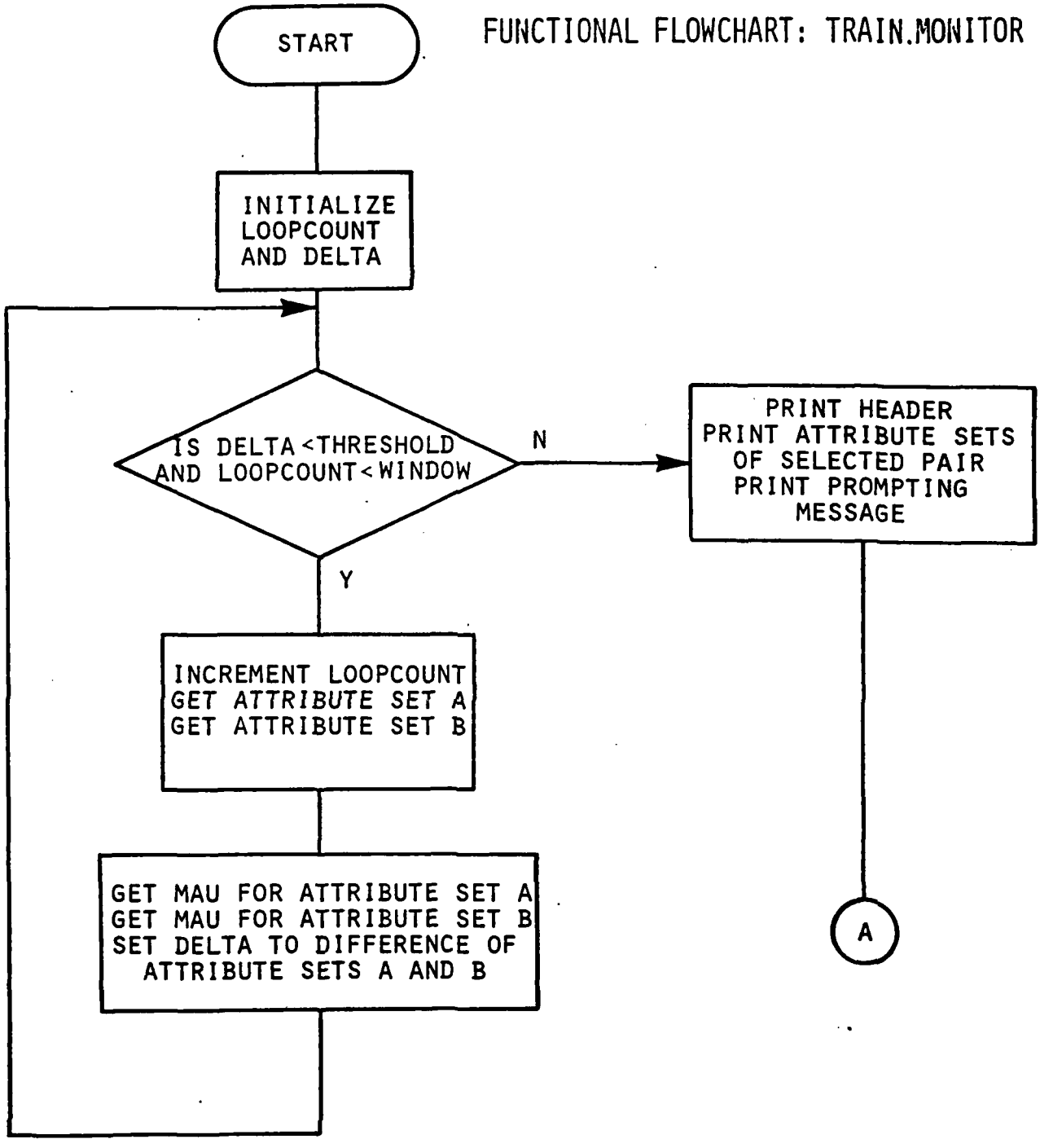
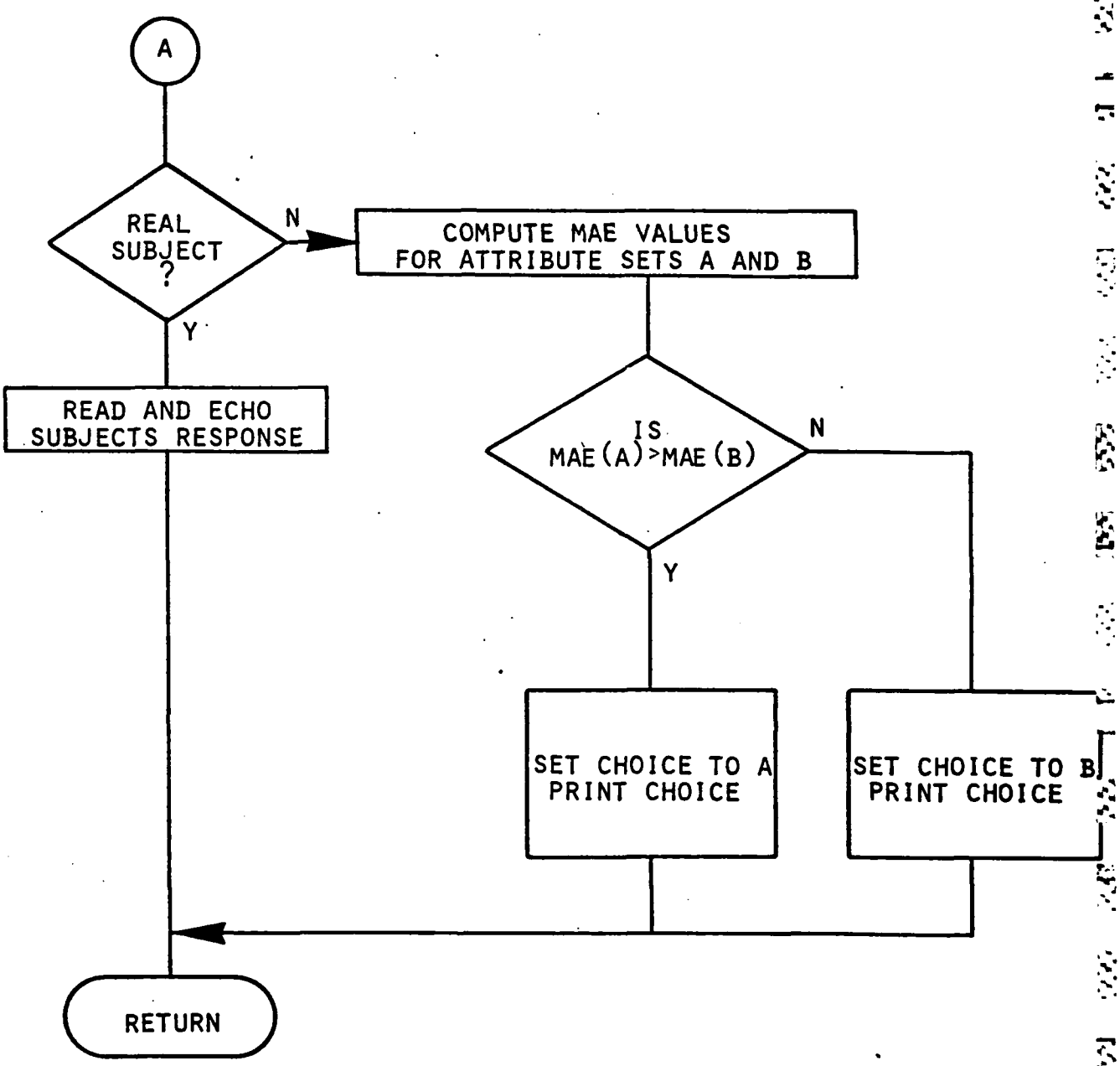

FUNCTIONAL FLOWCHART: TRAIN.MONITOR
START
INITIALIZE LOOPCOUNT AND DELTA
IS DELTA<THRESHOLD AND LOOPCOUNT<WINDOW
N
Y
PRINT HEADER
PRINT ATTRIBUTE SETS OF SELECTED PAIR
PRINT PROMPTING MESSAGE
INCREMENT LOOPCOUNT
GET ATTRIBUTE SET A
GET ATTRIBUTE SET B
GET MAU FOR ATTRIBUTE SET A
GET MAU FOR ATTRIBUTE SET B
SET DELTA TO DIFFERENCE OF ATTRIBUTE SETS A AND B
A

Figure A-3



FUNCTIONAL FLOWCHART: TRAIN MONITOR (CONTINUED) A-3

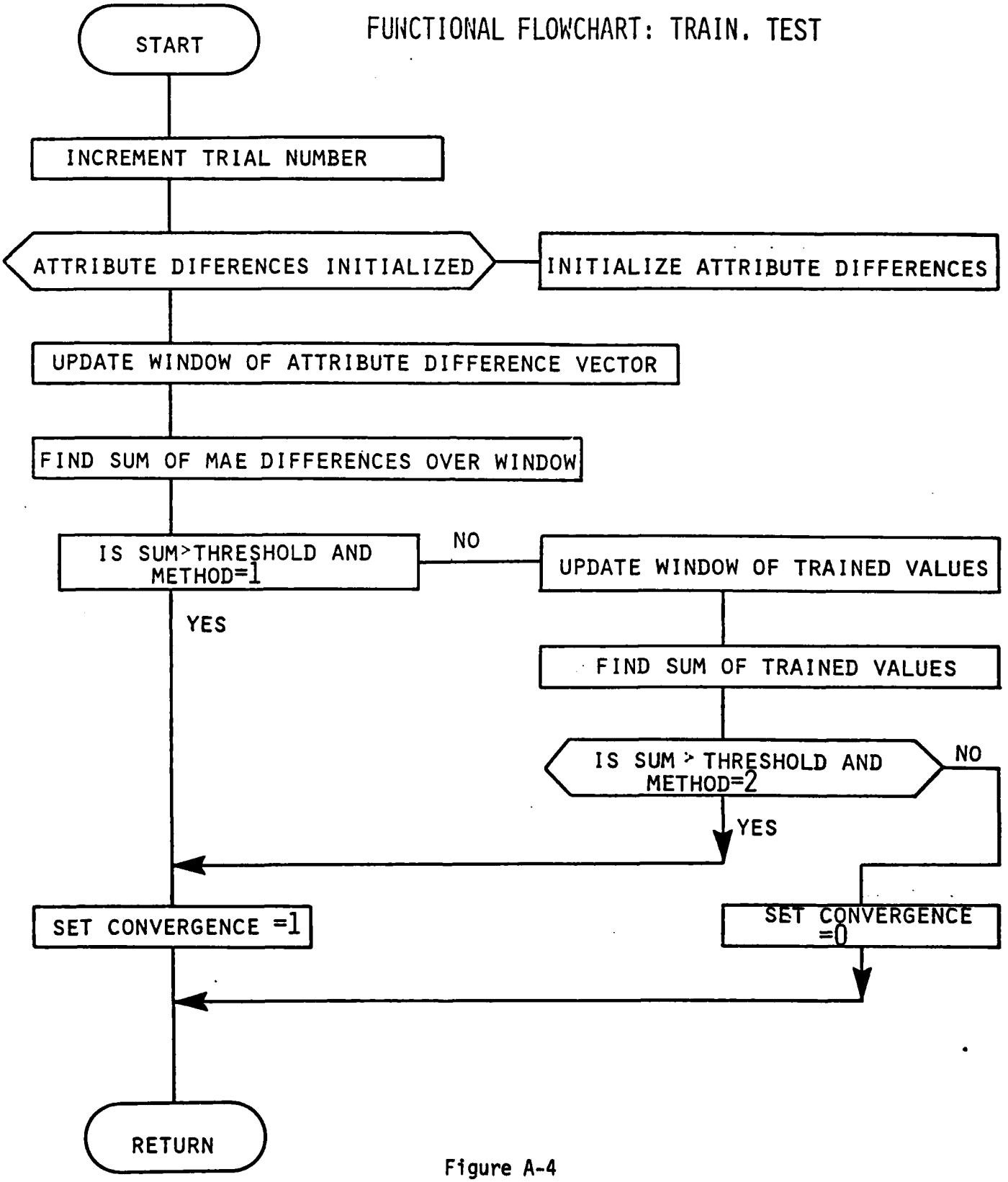

FUNCTIONAL FLOWCHART: TRAIN. TEST
START
INCREMENT TRIAL NUMBER
ATTRIBUTE DIFERENCES INITIALIZED
INITIALIZE ATTRIBUTE DIFFERENCES
UPDATE WINDOW OF ATTRIBUTE DIFFERENCE VECTOR
FIND SUM OF MAE DIFFERENCES OVER WINDOW
IS SUM>THRESHOLD AND METHOD=1
NO
YES
UPDATE WINDOW OF TRAINED VALUES
FIND SUM OF TRAINED VALUES
IS SUM > THRESHOLD AND METHOD=2
NO
YES
SET CONVERGENCE =1
SET CONVERGENCE =0
RETURN

Figure A-4

2.5 Genatt

This module returns to the calling program the newly chosen message attributes. In this version of the AIS model, the attributes of content area, age, familiarity, detail, priority and locale are randomly chosen.

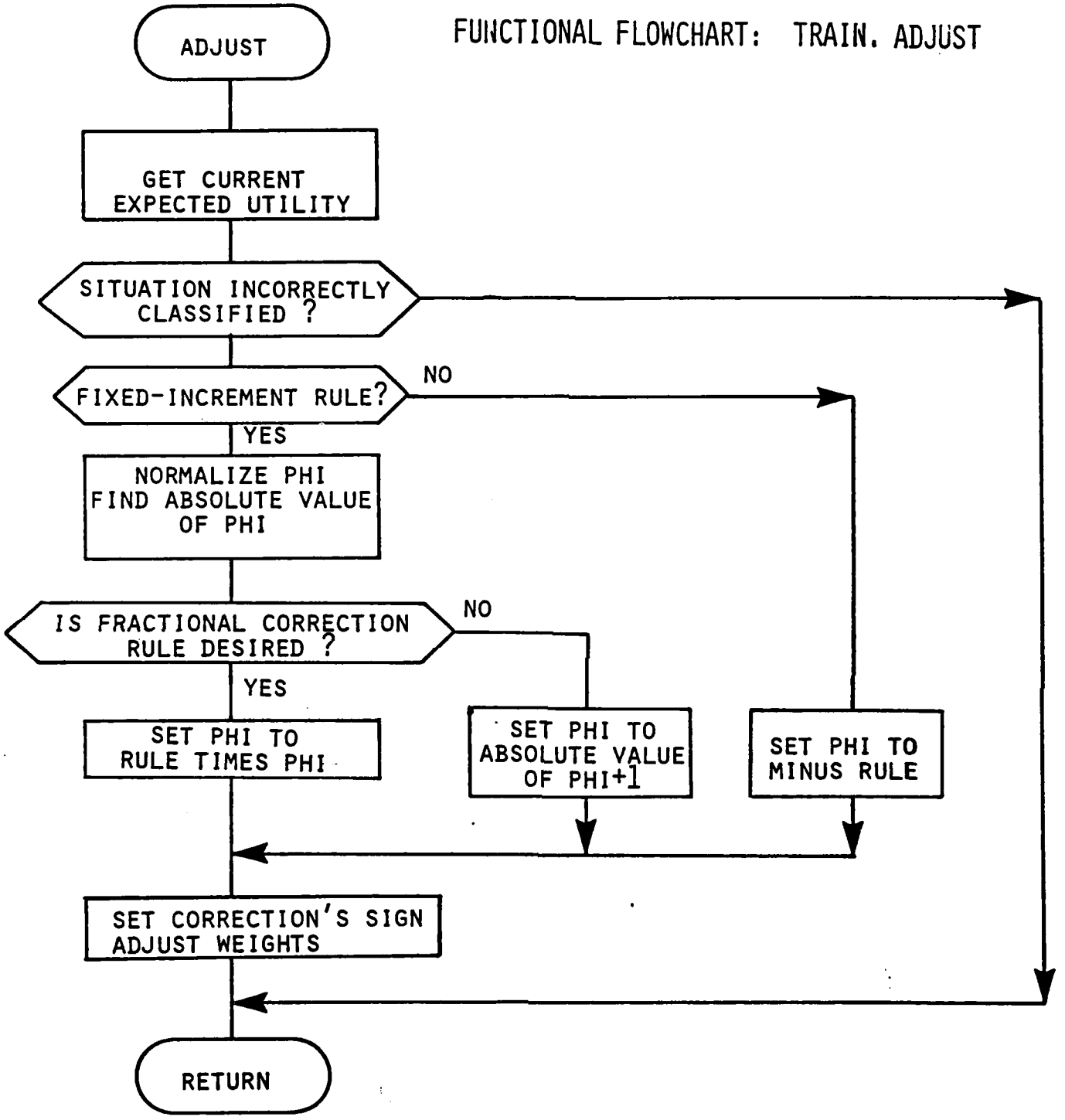

FUNCTIONAL FLOWCHART: TRAIN. ADJUST
ADJUST
GET CURRENT EXPECTED UTILITY
SITUATION INCORRECTLY CLASSIFIED ?
FIXED-INCREMENT RULE?
NO
YES
NORMALIZE PHI FIND ABSOLUTE VALUE OF PHI
IS FRACTIONAL CORRECTION RULE DESIRED ?
NO
YES
SET PHI TO RULE TIMES PHI
SET PHI TO ABSOLUTE VALUE OF PHI+1
SET PHI TO MINUS RULE
SET CORRECTION'S SIGN ADJUST WEIGHTS
RETURN

Figure A-5

FUNCTIONAL FLOWCHART: TRAIN.GENATT

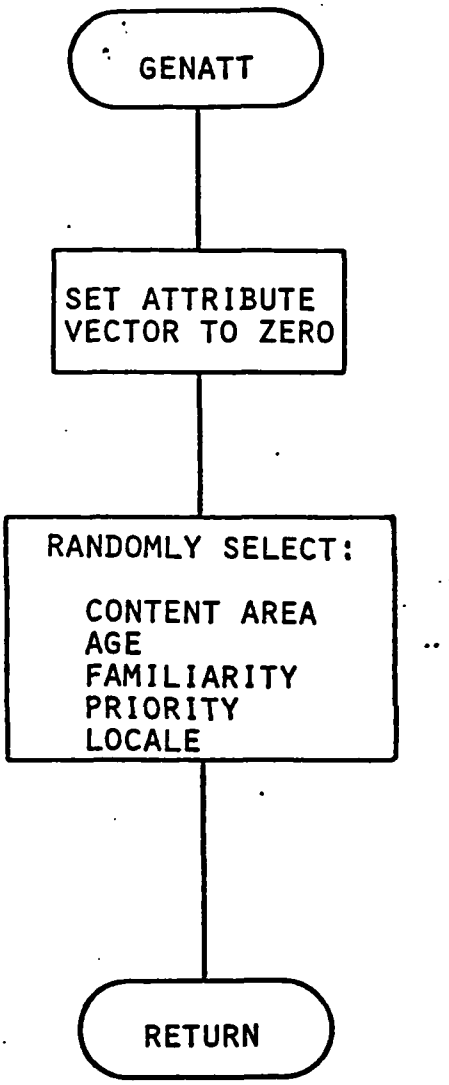


Figure A-6

SOURCE LIST FOR CALIBRATION PHASE

```
main()
PURPOSE:
        Master Scheduler for training subsystem.  A subject is
        presented with a pair of attribute vectors and is asked
        to select one.  Based on subject response a weight matrix is
        trained.  The process is repeated until conversence is
        obtained.
EXTERNALS:
        choice  -- subject's last choice
        sit     -- command situation
        weight  -- subject attribute weights
        mssatt  -- message attribute values
        diff    -- attribute difference vector
        trained -- 0=not trained;1=trained
        normliz()--Normalize weights
        monsubj()--Monitor subject
        trnmau()--Conditionally train current weights
        monsubj()--Monitor subject
EXIT CONDITIONS:
        Exits when weights converse.


printf("\033\105"); /*clear the screen                                    */
printf("           The scenario for this phase of the Adaptive Information\n");
printf("       Selector (AIS) exercise is taken from the TNG. You will\n");
printf("       be assigned a specific role or policy to follow. A pair\n");
printf("       of messages displayed by their specific attributes will\n");
printf("       be presented at your terminal.You are to select one of the\n");
printf("       messages (either A or B) based on your assigned role and the\n");
printf("       current command situation which is displayed along with the  \n");
printf("       message. Be careful to notice when the command situation \n");
printf("       changes.\n");
printf("           If you wish any further explanation on this training \n");
printf("       exercise please consult the training supervisor. The session \n");
printf("       will begin after you press the return key.\n");
/*   wait for carise return                                               */
```

```
/* normalize weisht matrix       */
        for(J=0;J<NSIT;J++)
                {

                wptr = &(weisht[sit][0]);
                normliz(wptr,wptr,NATT);
                }
 /*                                                          */
 for(cmdsit=0;cmdsit<NSIT;cmdsit++)
         {
         sit = cmdsit;
         /* If command is 1 set remainins weishts to cmd sit 0 */
         if(sit==1)
                 {
                 for(J=1;J<NSIT;J++)
                         {
                         for(i=0;i<NATT;i++)
                                 weisht[J][i] = weisht[0][i];
                         }
                 }
         convers = 0;

         trial = 0;
         while(convers==0)
                 {
                 n = NMSG-1;
                 monsubJ();              /*Monitor real or simulated
                                         subject response; clear screen */
```

```
/* Conditionally Train Current Weights   */

                for(i=0;i<NATT;i++)
                        {

                        diff[i] = mssatt[0][i]-mssatt[1][i];
                        }
                wptr = &(weisht[sit][0]);
                trained = trnmau(choice,NATT,wptr,diff,1.3);
                normliz(wptr,wptr,NATT);
if(trained>0.0001)
{
for(J=0;J<NSIT;J++)
        {
        printf(fileds,"\n");
        for(i=0;i<NATT;i++)
                printf(fileds," %3.2f",weisht[J][i]);
        }
}
/* At this point a conversence criterion must be met to exit from the loop*/
                convers = testcon();
                }
        }
/* Write weisht matrix on file   */
/* write weisht matrix to CRT   */
for(cmdsit=0;cmdsit<NSIT;cmdsit++)
        {
        printf(fileds,"\n");
        for(i=0;i<NATT;i++)
                printf(fileds," %3.2f",weisht[cmdsit][i]);
        }
cclose(fileds);

return;
}
```

```
nonsubj()

PURPOSE:
        Display attributes and obtain response from real (siml=0)

        or simulated (simul=1) subject.
EXTERNALS:
        simul--simulated subject flag.
        sel[]--Numbers of two messages to be presented
        choice--(0 or 1) choice of subject
        mssatt--message attributes
        simwt--weights for simulated subject
        dotp()--dot product routine
        sit--command situation
        attvec--attribute vector
        weisht--subject weisht vector
        mssatt--message attribute vector
        simwt--simulated subject weights

a = 0;
b = 1;
loopct = 0;
delta = 0.0;
while((delta<.01)&&(loopct<10))
        {
loopct++;
 senatt();
for(i=0;i<NATT;i++)
        mssatt[a][i] = attvec[i];
senatt();
for(i=0;i<NATT;i++)
        mssatt[b][i] = attvec[i];
rmau[0] = dotp(&(mssatt[a][0]),&(weisht[sit][0]),NATT);
rmau[1] = dotp(&(mssatt[b][0]),&(weisht[sit][0]),NATT);
delta = rmau[0]-rmau[1];
if(delta<0) delta = -1.0*delta;
        }
```

```
printf('\n\n');                    /*Clear screen*/
printf('\n COMMAND SITUATION: ');
if(sit==0) printf('PRE-MOBILIZATION');
if(sit==1) printf(' MOBILIZATION');
if(sit==2) printf(' COMBAT');
if(sit==3) printf(' REGROUP OPERATION');
/*write message header  */
/* Code to print words instead of attribute levels     */

printf('\nMESSAGE   CONTENT AREA       AGE         S/D
                                      FAMILIARITY   PRIORITY   LOCALE');


  for(i=0;i<2;i++)
          {
          if(i==0) printf('\n   A       ');
          if(i==1) printf('\n   B       ');
          if(mssatt[i][0]==1) printf('military        ');
          if(mssatt[i][1]==1) printf('intellisence    ');
          if(mssatt[i][2]==1) printf('nesotiations    ');
          if(mssatt[i][3]==1) printf('economics       ');
          if(mssatt[i][4]==0) printf('immediate');
          if(mssatt[i][4]==1) printf('recent    ');
          if(mssatt[i][4]==2) printf('old       ');
          if(mssatt[i][5]==0) printf('   summary ');

          else printf('   detailed');
          if(mssatt[i][6]==0) printf('   unseen         ');

          else printf('   seen           ');
          if(mssatt[i][7]==0) printf('low        ');
          else printf('high       ');
          if(mssatt[i][8]==0) printf('own');
          else printf('adjacent');
          }
  /*
  printf('\nSelect A or B');
```

```
printf("\n");
if(simul == 1)
        {
        mau[0] = dotp(&(mssatt[a][0]),&(simwt[sit][0]),NATT);

        mau[1] = dotp(&(mssatt[b][0]),&(simwt[sit][0]),NATT);
        if(mau[0]>mau[1])
                {
                choice = 1;
                printf("\nA");
printf("\nchoice= %d",choice);
                }
                else
                {
                choice = 2;
                printf("\nB");
printf("\nchoice= %d",choice);
                }
return;
        }
/* Real Subject                                          */
/* Assemble any keyboard input locally with no read hangins.    */
out = 0;

while(out !=1)
{
kbfptr = kbuf;
while(!empty(crtfd))
       read(0,kbfptr,1);        /* Read next char           */
       xpos = kbfptr-kbuf;      /*Get horiz shift for echo to CRT*/
       temp = *kbfptr;

       kbfptr++;                /*Update local buffer pointer*/
       if((xpos++)==BUFFER_LENGTH)  kbfptr--;
```

```
/* If a carriage return was typed, process command*/
        if(temp==NEWLINE)
                {
                switch(kbuf[0])
                        {
                        case'A':
                        choice = 1;
                        break;
                        case 'B':
                        choice = 2;
                        break;
                        }
                out = 1;
                }
        }
}
printf("\return from monsubj, out= %d",out);
        return;
}
```

```
testcon()

PURPOSE:
        Updates parameters for conversence testing according to two
        different methods.  When conversence is met according to the
        specified criteria a one is returned indicating conversence;
        otherwise a zero is returned.
EXIT CONDITIONS:
        A parameter indicating conversence is returned.
ALGORITHMS:
        1--Method 1 Fast Conversence Test--Behavior over the
           window is recreated and evaluated with the new


           weights.
        2--Method 2 Slow Conversence Test--The amount of training

           over the threshold is accumulated and evaluated.
EXTERNALS:
        cnvwnd1=Conversence window for method 1
        cnvwnd2=Conversence window for method 2
        method=1 for fast method; 2 for slow method
        thold1=Threshold for method 1
        thold2=Threshold for method 2
        attdif[][]=Saved attribute difference vectors over cnvwnd1
        choice=Last choice=0 or 1
        diff[]=attribute difference vector
        dotp()=dot product function
        weisht[][]=weights
        sit=current command situation
        trnsav=window of saved values of trained
        chosen=window of saved values of chosen
        trained-- 0=not trained;1=trained
        attdif--window of values of difference
```

```
cnvwnd1 = WIND1;
trial++;
if(trial==1)
        {
        for(j=0;j<NATT;j++)
                {for(i=0;i<cnvwnd1;i++)
                        attdif[i][j] = 10.0;
             } }
convers = 0;
/*Method 1 Fast Conversence Test  */
/*     Move back values    */
for(i=1;i<cnvwnd1;i++)
        {
        k=i-1;
        chosen[k] = chosen[i];
        for(j=0;j<NATT;j++)
                {
                attdif[k][j] = attdif[i][j];
                }
        }
                chosen[cnvwnd1-1] = choice;

        for(j=0;j<NATT;j++)
                attdif[cnvwnd1-1][j] = diff[j];
/*     Calculate new accumulated maudif          */
sum = 0.;
if(trial>WIND1)
{
for(i=0;i<cnvwnd1;i++)
        {
        maudif = dotp(&attdif[i][0],&(weight[sit][0]),NATT);
        if((maudif>0)&&(chosen[i]==2)) sum = sum+maudif;
        if((maudif<0)&&(chosen[i]==1)) sum = sum-maudif;
        }
if((sum<thold1)&&(method==1)) convers = 1;
printf("\nMaudif for fast method= %8.4f",sum);
}
```

```
/*                                                              */
/*Slow Conversence test                                        */
for(i=1;i<cnvwnd2;i++)
        trnsav[i-1] = trnsav[i];
trnsav[cnvwnd2-1] = trained;
if(trial>WIND2)
{
sum = 0.;
for(i=0;i<cnvwnd2;i++) sum = sum+trnsav[i];
if((sum<thold2)&&(method==2)) convers = 1;
printf("\nsum of trained for slow method= %5.4f",sum);
}
/*                                                              */
printf("\nend of testcon; convers= %d",convers);
return(convers);
}
```

```
trnmau(ia,n,w,x,rule)

PURPOSE:
  Trains a multi-attribute, utility model.  The model is
  treated as a threshold logic machine with no constant
  term and a specified correction rule is applied.

ARGUMENTS:
  ia      - observed category  (integer)
  n       - no. of components in x  (integer)
  w       - model vector  (real)
  x       - attribute-difference vector  (real)
  rule    - correction rule  (real)
            (sussested:  rule=1.3)

EXIT CONDITIONS:

  When appropriate, the model is trained and the adjustment
  constant is returned.  Zero is returned when no training
  takes place.

ALGORITHMS:
  When EU(x)<=0, and observed category is 1,
      (1)  w=w+c*x;
  When EU(x)>=0, and observed category is 2,
      (2)  w=w-c*x.

      Fixed-Increment Rule:
        c is a constant > 0.
      Absolute-Correction Rule:
        c=si(abs((w)dot(x))/((x)dot(x)),
        and si is the greatest-integer function.
      Fractional-Correction Rule:
        c=lambda*abs((w)dot(x))/((x)dot(x)).
```

```
        adj=0.;         /*initialize adjustment record.*/

        phi=dotp(w,x,n); /*Get current EU.              */
/*    Train only if situation is incorrectly classified.  */
        if (((phi<=0.)||(ia!=1))&&((phi>=0.)||(ia!=2))&&
            (dotp(x,x,n)!=0.0))
/*      Use  desired adjustment rule to set adjustment.   */
        {  if (rule<0.)
              phi= -rule; /*Fixed-Increment             */
           else
           {  phi=phi/dotp(x,x,n);
              if (phi<0.)
                 phi= -phi;
              if (rule==0.)
/*                Absolute-Correction                  */
                 phi=abs(phi+1.);
/*                Fractional-Correction                */
              else  phi=rule*phi;
           }
           adj=phi;

           if (ia!=1)    /*Set correction's sign.       */
              phi= -phi;
/*      Adjust weights.                                          */
           for (i=0;i<n;i++)
              w[i]=w[i]+phi*x[i];
        }
     return(adj);
}
```

```
        statt2()

/*
NAME:
 statt2.c
FUNCTION:
 Gets attributes of all messages and the type of each message
ALGORITHM:
 for (i = 0, number of messages)
 {
    find message number
    find message type
    find message end
    extract message attributes
    find double slashes
 }
 return
RETURNS:
 a 0 if successful or 2 if an error occurred
GLOBALS:
 mssfd            -  message file descriptor                  (integer)
 mssfl[]          -  full path name of message file           (character)
 typem[]          -  vector containing message types          (integer)
 mssatt[61][9]    -  message attributes                       (float)
 msatt[61][2][9] -  message attributes for each subject      (float)
```

```
/*    Open message file.                                              */
          mssfd=copen(mssfl,'r');
/*    Get each message's attributes.                                  */
          for (i=0;i<nmess;i++)
          {  temprc='2';
             while ((temprc!='0')&&(temprc!='1'))     /*  Find mss. no. */
                temprc=csetc(mssfd);

             while (temprc!='/')                  /*  Find mss. type.    */
             {  temp2c=temprc;


                temprc=csetc(mssfd);
             }
/*    Record message type.                                            */
             switch (temp2c)
             {
             case 'M':
                j=0;
                break;
             case 'I':
                j=1;
                break;
             case 'N':
                j=2;
                break;
             case 'E':
                j=3;
                break;
             default:
                j=0;
                break;
             }
```

```
                typem[i]=J;
                tempc = '0';
                while (tempc != '/')      /*     find mss. end     */
                   tempc = csetc(mssfd);
                for (J=0;J<natt;J++)
                {  test = scanf(mssfd,"%f%c",&mssatt[i][J],&tempc);
                   msatt[i][0][J] = msatt[i][1][J] = mssatt[i][J];
                   if (test != 2)
                      goto error;
                }
                temp2c = '1';
                while (temp2c != '/')       /*  set block terminator  */
                {  tempc = csetc(mssfd);
                   if (tempc == '/')
                      temp2c = csetc(mssfd);
                }
             }
          cclose(mssfd);
          return(0);
          cclose(mssfd);
 error:   return(2);
}
```

APPENDIX B
VERIFICATION PHASE

MODEL CALIBRATION PHASE

1. Introduction

The emphasis of the testing phase is to provide a framework to test the validity of the AIS model. Toward this end, two sets of statistics will be gathered. These statistics provide a measure of the utility of the AIS model in routing and prioritizing messages in the TCO environment. Using the two commands available to the user during this phase, implicit and explicit declaration of the value of a multi-attributed message can be obtained. When the user enters the next message command, the rating which follows provides an explicit value of the message utility. In the alternate case, when the user enters the select message command, it is assumed that the message selected by the user has the highest utility to the user. From this, an implicit value of the message utility can be obtained.

2. Principal Functions

The modules which make up the testing phase of the AIS model are:

(1)	Main	-	Execution Controller
(2)	Route	-	Message Router
(3)	Monitor	-	User Monitor
(4)	Process	-	Command Interpreter

Figure B-1 shows the precedence of the modules in the calibration phase. Route and monitor are routines called by main; process is a subroutine called by monitor.

2.1 Main

The flowchart for this module is shown in Figure B-2. This module is responsible for control of program execution. As described in the users

guide, three control cases, each with four command situations, is run for each user. Within each command situation, a process of routing and monitoring is repeated. If no message is to be routed, the cycle consists only of monitoring the subjects.

At the conclusion of each command situation, the statistics gathered are printed, and the next command situation initiated.

2.2 Route

Routing of a message consists of determination of the recipient, logging the message in the selected users bin, and showing the user the new updated bin. The functional flowchart is shown in Figure B-3. The selected message recipient depends on the control case, and the calculated MAE for the message. Logging of a message consists of updating the users bin pointer with the message ID, and the calculated MAE value of the message.

2.3 Monitor

This module is responsible for monitoring keyboard activity of each user in the system. This is accomplished by reading the characters that are typed by a user, and processing the resulting command when it is complete. The functional flowchart is given in Figure B-4.

2.4 Process

A description of the user control commands was given in an earlier section; they are repeated here to aid in describing the function of this module. In the case of a 'halt' or 'proceed' command, the required processing is straight forward. When the user enters 'proceed', the command situation is updated and the program continues.

The functional flowchart given in Figure B-5 shows that the system next

checks to see if a user rating has been entered. The system currently allows the user to choose a message from the bin, or allow the AIS model to choose a message for the user. If the AIS model chooses a message, the user is required to enter a rating. A functional flowchart for the selection module which is called by this process is given in Figure B-6.

In the selection module, the commands 'next' and 'select' are implemented with the error messages that may result from each. Finally, the message that is chosen, whether by the model or by the user, is displayed on the user's terminal. At this time, the user's bin is also updated to show that the message no longer resides in his bin.

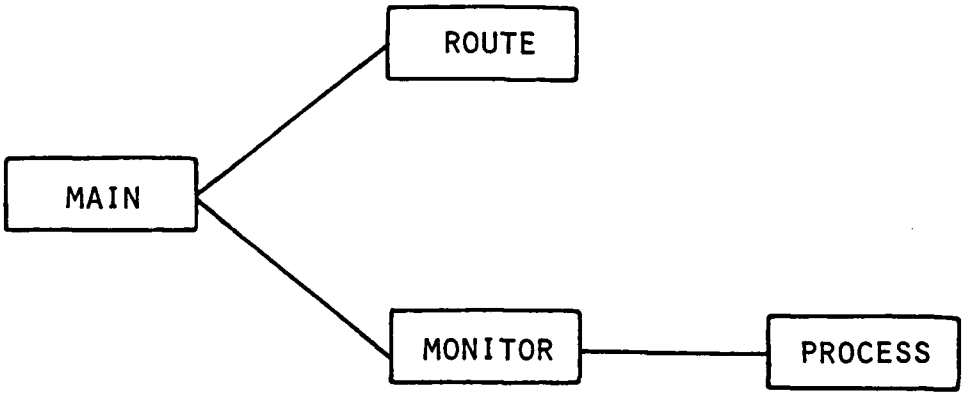


FIGURE B.1 MODULE INVOKING SEQUENCE FOR TESTING PHASE

FUNCTIONAL FLOWCHART: TEST. MAIN 1

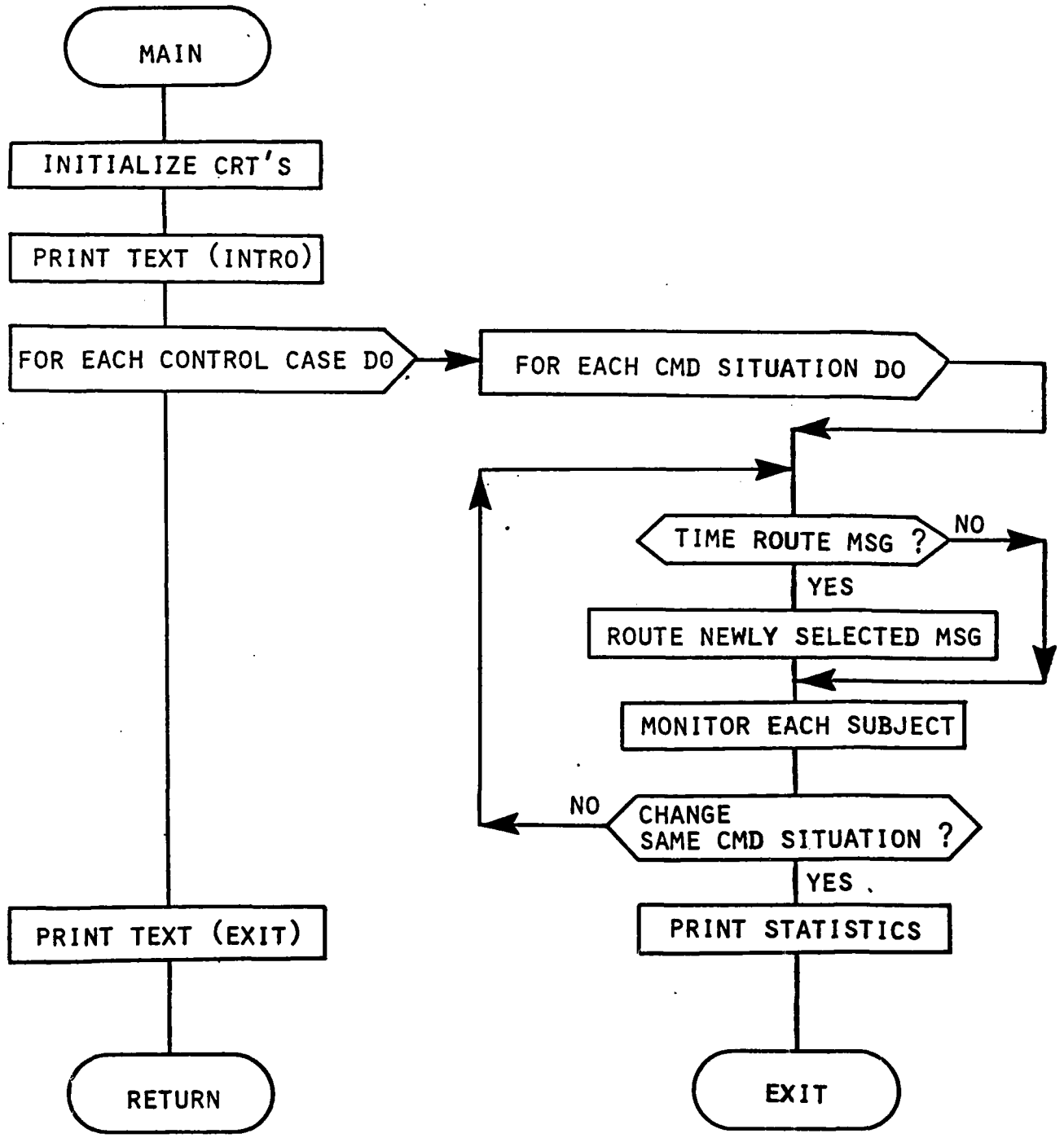


Figure B.2

FUNCTIONAL FLOWCHART OF: TEST.ROUTE 2

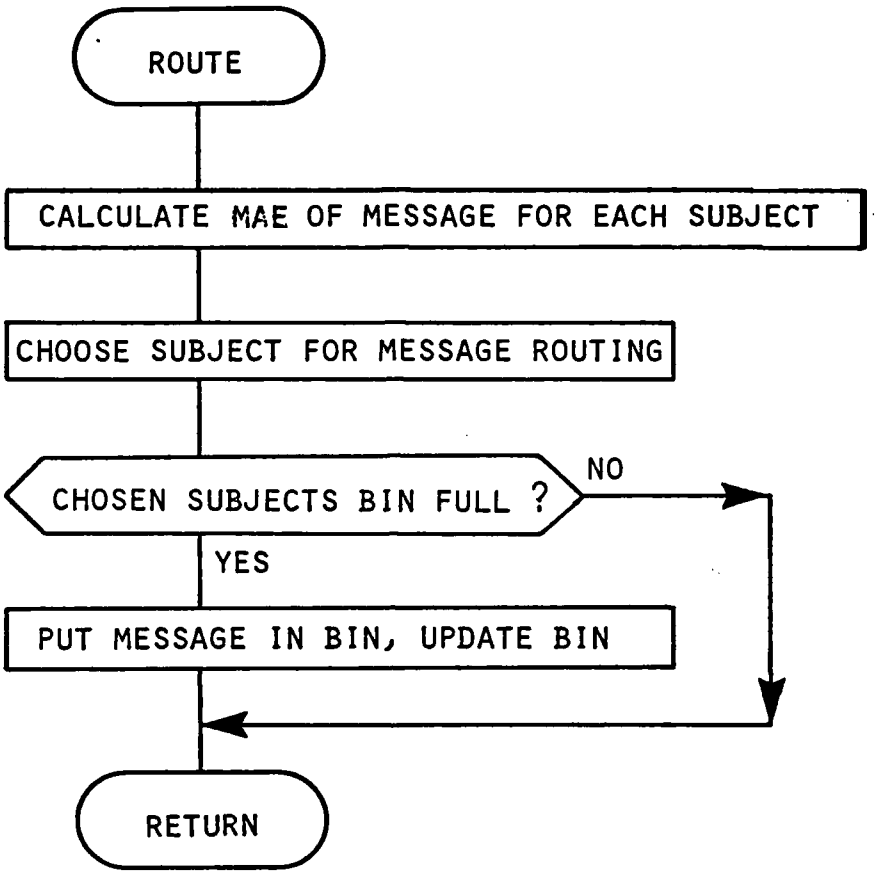


Figure B.3

FUNCTIONAL FLOWCHART: TEST.MONITOR 3

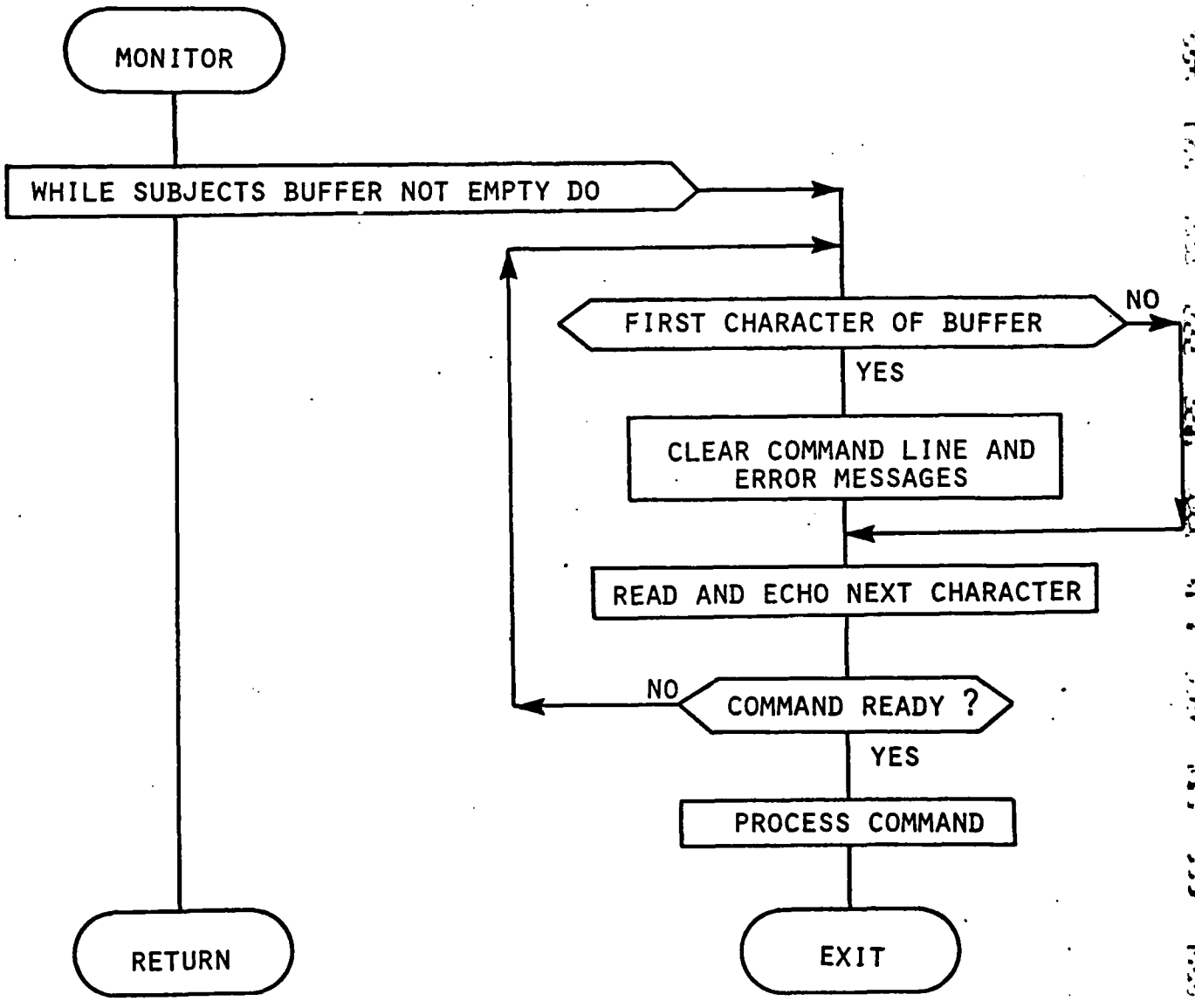


Figure B.4

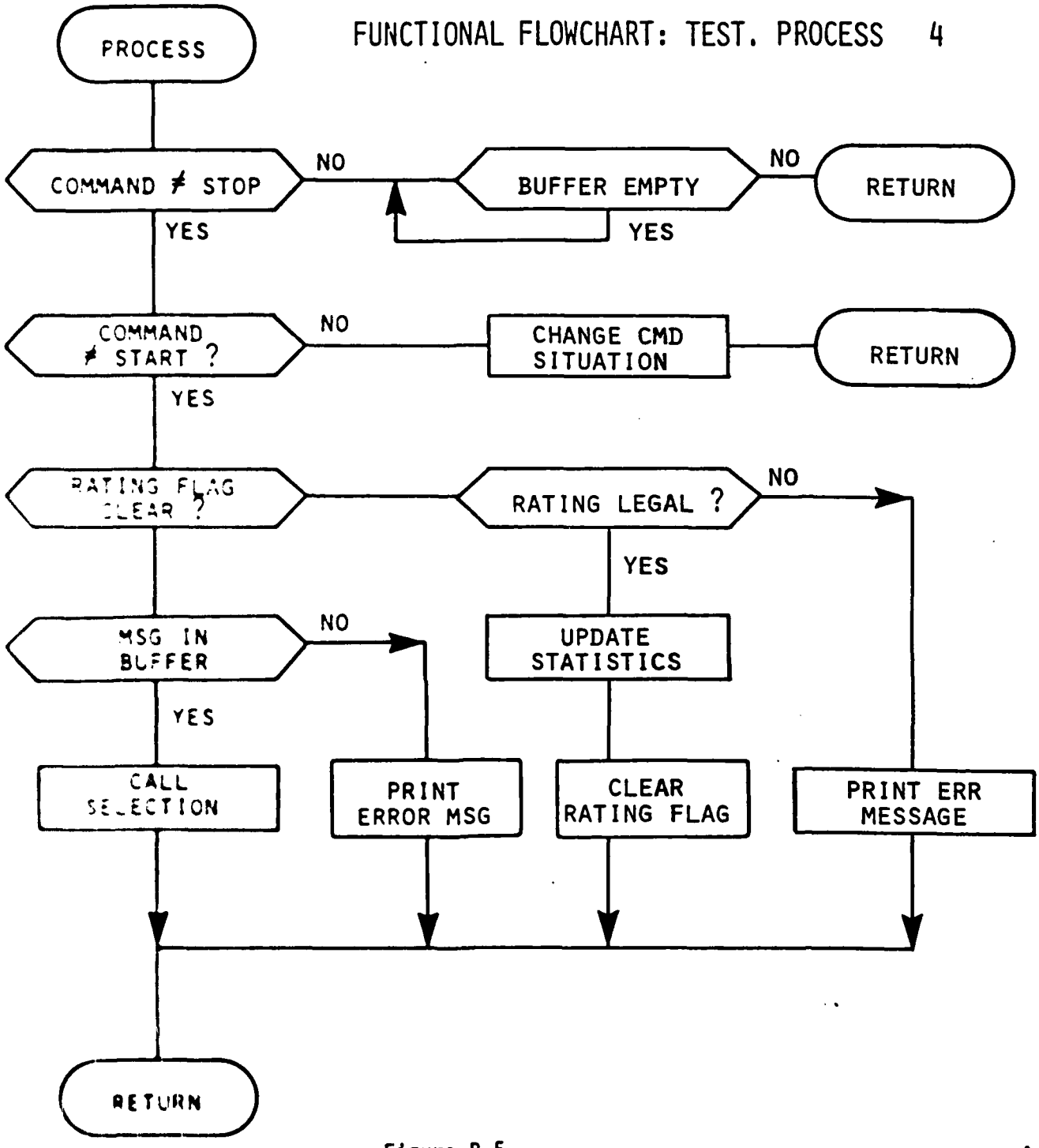

FUNCTIONAL FLOWCHART: TEST. PROCESS 4
PROCESS
COMMAND ≠ STOP
NO
BUFFER EMPTY
NO
RETURN
YES
YES
COMMAND ≠ START ?
NO
CHANGE CMD SITUATION
RETURN
YES
RATING FLAG CLEAR ?
RATING LEGAL ?
NO
YES
MSG IN BUFFER
NO
UPDATE STATISTICS
YES
CALL SELECTION
PRINT ERROR MSG
CLEAR RATING FLAG
PRINT ERR MESSAGE
RETURN

Figure B.5

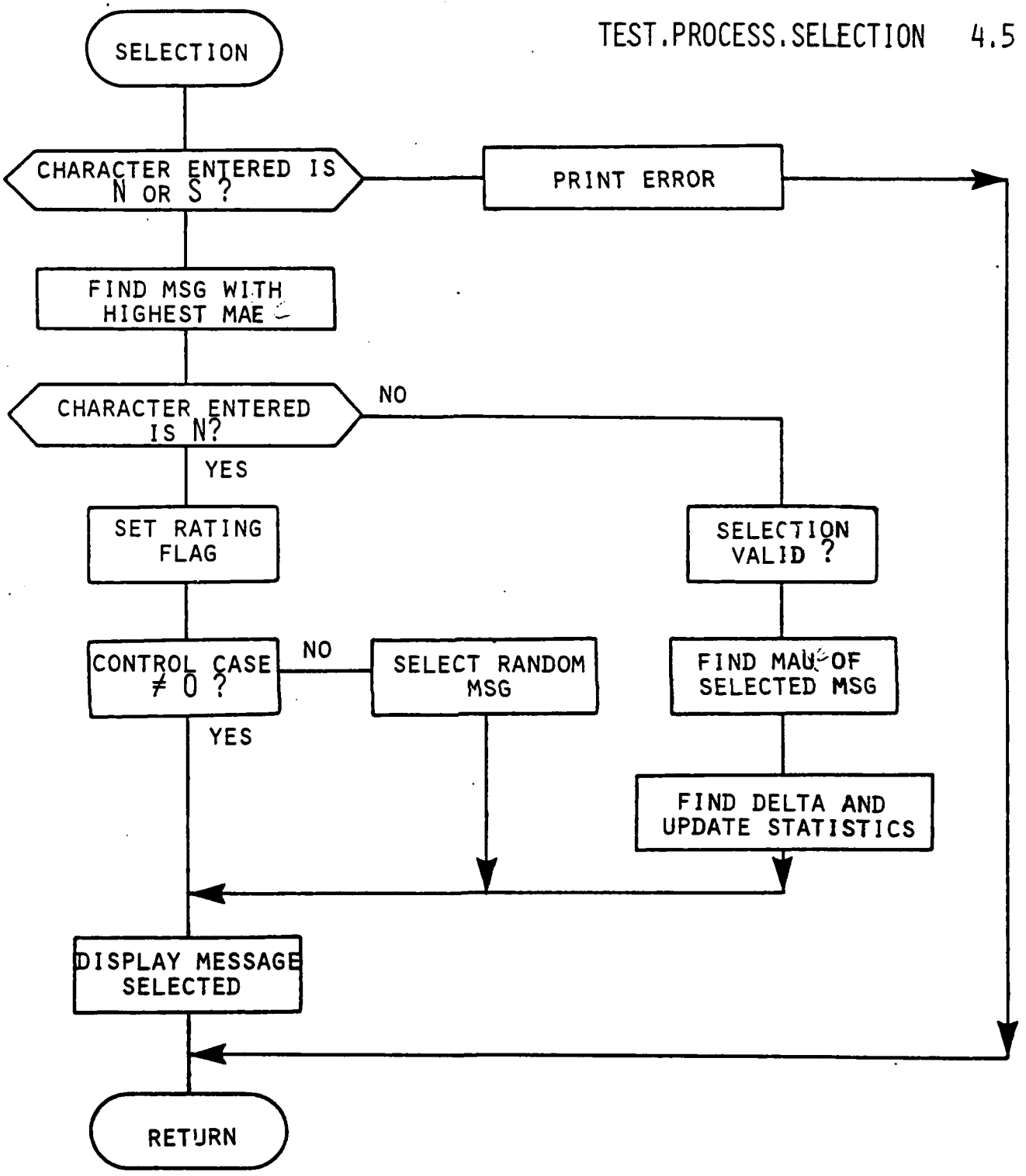

SELECTION
CHARACTER ENTERED IS N OR S ?
PRINT ERROR
FIND MSG WITH HIGHEST MAE
CHARACTER ENTERED IS N?
NO
YES
SET RATING FLAG
SELECTION VALID ?
CONTROL CASE ≠ 0 ?
NO
SELECT RANDOM MSG
FIND MAU OF SELECTED MSG
YES
FIND DELTA AND UPDATE STATISTICS
DISPLAY MESSAGE SELECTED
RETURN

Figure B.6

SOURCE LISTING OF TESTING PHASE

```
      main()

/*
NAME:
 exec.c
FUNCTION:
     This is the main program which loops through the control
 cases and command situations. It calls route.c to send messages
 to a number of subjects at preset intervals. It also calls
 monitor.c to accept each subject's responses. It prints accumulated
 statistics from subject's responses after each control case
 and command situation.
ALGORITHM:
 initialize CRT's and open files
 read all messages attributes
 clear screen
 for (control case = 0;ncc)
 {
    while (command situation < ncs)
    {
       route messages
       monitor subjects
       if (time for next command situation) print statistics
    } end of while loop
 } end of for loop
 end
```

```
GLOBALS:
 mssfd          -  message file descriptor                    (integer)
 crtfd[]        -  CRT file descriptor for each subject       (integer)
 mssfl[]        -  full path name of message file             (character)
 kbuf[2][7]     -  command buffer area                        (character)
 *kbufptr[]     -  buffer area pointer                        (character pointer)
 echo[]         -  character echo buffer                      (character)
 com[]          -  character string COMMAND:                  (character)
 clr[]          -  code to clear screen                       (character)
 tcs[]          -  timing vector for each command situation   (integer)
 vect[]         -  timing vector for each message             (integer)
 ch             -  flag to change command situation           (integer)
 avs[]          -  average rating of each subject             (float)
 var[]          -  variance of ratings of each subject        (float)
 n[]            -  total number of ratings for each subject   (float)
 cc             -  current control case                       (integer)
 csl            -  current command situation                  (integer)
 ib[]           -  pointer to top of each subject's bin       (integer)
 mau[2][10]     -  mau value of each message in each          (float)
                   subject's bin
 hit[]          -  counter for matches between                (integer)
                   system and subject selected messages
 tot[]          -  total number of selected messages          (integer)
 totd[]         -  total of mau differences between subject   (float)
                   and system selections
 prand()        -  random number generator [0,1)             (float)
CALLS:
 open,crtset,write,stbias2,printf,statt2,infam0,copen,sleep,tmeasur,prand,
```

```
for(cc = icc;cc < ncc;cc++)    /* control case loop.                          */
   {
    while(csl < ncs )   /* command situation loop.                          */
         {
          sleep (nsecond);/* sleep n second.                                 */
          t1 = tmeasur(); /* recored initialize time.                       */
          clock = clock + nsecond; /* update the clock.                      */
          if(vect[p] <= clock)  /* time for next message.                   */
            {
             p++;  /* increment p by 1.                                     */
             temp = prand()*nmess;
             inrmess = temp;
             area(inrmess);   /* assign attribute no 8.                     */
             route(inrmess);  /* call route routine.                        */
            clock=0;
            }
          monitor(0);
          monitor(1);
          t2 = tmeasur();  /* record end time.                              */
          t3 = (t2-t1)/60. + clock;  /* time interval.                      */
          t4 = t4+t3;
          if((t4 >= tcs[csl])||(ch==1))  /* if time to change command situation. */
            {
            write(crtfd[0],chgsit,34);
            write(crtfd[1],chgsit,34);
            sleep(3);
            for(i=0;i<=1;i++)
            {  write(crtfd[i],clr,2);
               prthd(i);
            }
            write(crtfd[0],stdby,19);
            write(crtfd[1],stdby,19);
```

```
            if(n[0]!=0.) avgd[0] = totd[0]/n[0];
            else avgd[0] = 0.;
            if(n[1]!=0.) avgd[1] = totd[1]/n[1];
            else avgd[1] = 0.;
           if(tot[0]!=0.) phit[0] = hit[0]/tot[0];
           else phit[0] = 0.;
          if(tot[1]!=0.) phit[1] = hit[1]/tot[1];
          else phit[1] = 0.;
/*              print statistics.                                          */
     printf("\nfor control case %d, and command situation %d,\n",cc,csl);
 printf("\nsubj 0: mean %f, var %f, ratio agree %f, avg dist %f",
       avg[0],var[0],phit[0],avgd[0]);
 printf("\nsubj 1: mean %f, var %f, ratio agree %f, avg dist %f",
       avg[1],var[1],phit[1],avgd[1]);

            csl++;   /* next command situation.                          */
            ch = 0;
            clock = 0;  /* reset statistics and clock.                   */
             t4 = 0;    /* initialize accumulate time to 0.              */
             p = 0;     /* initialize counter of time interval to 0.     */
             n[0] = n[1] = 0. ;
             avg[0] = avg[1] = 0. ;
             var[0] = var[1] = 0. ;
             ib[0] = -1; ib[1] = -1;
             hit[0] = hit[1] = 0. ;
             tot[0] = tot[1] = 0. ;
             for (i=0;i<10;i++)
             { mau[0][i] = 0.; mau[1][i] = 0.; }
            }  /* end of if loop.                                         */


         }  /* end of while loop.                                         */
      csl = 0;
      infam0();  /* initialize fimilarity vector to 0.                   */

   } /* end of control case loop.                                         */
```

```
  route(mssid)

/*
NAME:
 route.c
FUNCTION:
     Distribute incoming message to appropriate subject's bin by either
 using mau or randomly
ALGORITHM:
 Calculate mau of message for each subject
 if(control case = 2) place message in bin of subject with highest mau value
 otherwise place randomly in one of the subject's bin
 return
PARAMETERS:
 mssid            -  message number                              (integer)
GLOBALS:
 cc               -  current control case                        (integer)
 crtfd[]          -  CRT file descriptor for each subject        (integer)
 ib[]             -  pointer to top of each subject's bin        (integer)
                     of messages
 csl              -  current command situation                   (integer)
 scr              -  subject chosen for routing                  (integer)
 echo[]           -  character echo buffer                       (character)
 msatt[61][2][9]  -  message attributes of all messages for      (float)
                     each subject
 weight[2][4][9]  -  weights for each subject under all          (float)
                     command situations
 bin[2][10]       -  stack containing current collection of      (integer)
                     messages for each subject
 mau[2][10]       -  mau value of each message in each           (float)
                     subject's bin
 dotp()           -  dot product routine                         (float)
```